

TITLE OF THE INVENTION

**SOUND SOURCE SYSTEM BASED ON COMPUTER SOFTWARE AND
METHOD OF GENERATING ACOUSTIC WAVEFORM DATA**

BACKGROUND OF THE INVENTION

The present invention relates to a sound source system that combines music tone waveform generating modules made of software, and that generates music tone waveform data based on music tone waveform generating computation performed by each music tone waveform generating module. In addition, the present invention relates to a sound source waveform generating method that uses a general-purpose computation processing machine for executing a waveform computation algorithm so as to generate tone waveform data.

Conventionally, in order to generate a music tone according to a variety of music tone generating methods such as a waveform memory tone generating method and an FM tone generating method, a circuit for implementing the music tone generating method is constituted by dedicated hardware such as an LSI specifically designed for a sound source and a digital signal processor (DSP) that operates under the control of a fixed microprogram. The music tone generator constituted by the dedicated hardware is generically referred to as a hardware sound source hereafter. However, the hardware sound source requires dedicated hardware components, hence reduction of the product cost is difficult. It is also difficult for the hardware sound source to flexibly modify its specifications once the design has been completed.

Recently, as the computational performance of CPU has been enhancing, tone generators have been developed in which a general-purpose computer or a

1 CPU installed on a dedicated tone generator executes software programs
2 written with predetermined tone generation processing procedures to generate
3 music tone waveform data. The tone generator based on the software programs
4 is generically referred to as a software sound source hereafter.

5 Use of the hardware sound source in a computer system or a computer-
6 based system presents problems of increasing the cost and decreasing the
7 flexibility of modification. Meanwhile, the conventional software sound
8 sources simply replace the capabilities of the dedicated hardware devices such
9 as the conventional tone generating LSI. The software sound source is more
10 flexible in modification of the specifications after completion of design than
11 the hardware sound source. However, the conventional software sound source
12 cannot satisfy a variety of practical demands occurring during vocalization or
13 during operation of the sound source. These demands come from CPU
14 performance, system environment, user preferences and user settings. To be
15 more specific, the conventional software sound sources cannot satisfy the
16 demands for changing fidelity of an outputted music tone waveform (not only
17 the change to higher fidelity but also to lower fidelity) and demands for
18 changing the degree of timbre variation (for example, change from normal
19 timbre variation to subtle timbre variation or vice versa).

20 Recently, an attempt has been made to generate tone waveform data by
21 operating a general-purpose processor such as a personal computer to run
22 software programs and to convert the generated digital tone waveform data
23 through a CODEC (coder-decoder) into an analog music tone signal for
24 vocalization. The sound source that generates the tone waveform data by such
25 a manner is referred to as the software sound source as mentioned before.
26 Otherwise, the tone waveform data may be generated by an LSI dedicated to

1 tone generation or by a device dedicated to tone generation having a digital
2 signal processor (DSP) executing a microprogram. The sound source based on
3 this scheme is referred to as the hardware sound source as mentioned before.

4 Generally, a personal computer runs a plurality of application software
5 programs in parallel. Sometimes, a karaoke application program or a game
6 application program is executed concurrently with a software sound source
7 application program. This situation, however, increases a work load imposed
8 on the CPU (Central Processing Unit) in the personal computer. Such an over
9 load delays the generation of tone waveform data by the software sound source,
10 thereby interrupting the vocalization of a music tone in the worst case. When
11 the CPU is operating in the multitask mode, the above-mentioned concurrent
12 processing may cause the tasks other than the tone generation task into a wait
13 state.

14 In the hardware sound source, a waveform computation algorithm is
15 executed by the DSP or the like to generate tone waveform data. The
16 performance of the DSP for executing the computation has been enhanced
17 every year, but the conventional tone waveform data generating method cannot
18 make the most of the enhanced performance of the DSP.

20 SUMMARY OF THE INVENTION

21 It is therefore an object of the present invention to provide a sound
22 source system based on computer software capable of reducing cost by
23 generating a music tone by a software program without adding special
24 dedicated hardware and, at the same time, capable of changing the load of a
25 computation unit for computing music tone waveform and improving the
26 quality of an output music tone.

1 It is another object of the present invention to provide a tone waveform
2 data generating method that is capable of generating tone waveform data
3 without interrupting the vocalization of a music tone even if the CPU load is
4 raised high, and capable of, when the CPU is operating in the multitask mode,
5 processing tasks not associated with the tone waveform generation without
6 placing these tasks in a wait state.

7 It is still another object of the present invention to provide a tone
8 waveform data generating method that makes a hardware sound source fully
9 put forth its computational capability to provide the waveform output having
10 higher precision than before.

11 The inventive sound source apparatus has operation blocks composed of
12 softwares used to compute waveforms for generating a plurality of musical
13 tones through a plurality of channels according to performance information. In
14 the inventive apparatus, a setting device sets an algorithm which determines a
15 system composed of selective ones of the operation blocks systematically
16 combined with each other to compute a waveform specific to one of the musical
17 tones. A designating device responds to the performance information for
18 designating one of the channels to be used for generating said one musical tone.
19 A generating device allocates the selective operation blocks to said one channel
20 and systematically executes the allocated selective operation blocks according
21 to the algorithm so as to compute the waveform to thereby generate said one
22 musical tone through said one channel.

23 Preferably, the setting device sets different algorithms which determine
24 different systems corresponding to different timbres of the musical tones. Each
25 of the different systems is composed of selective ones of the operation blocks

1 which are selectively and sequentially combined with each other to compute a
2 waveform which is specific to a corresponding one of the different timbres.

3 Preferably, the setting device comprises a determining device that
4 determines a first system combining a great number of operation blocks and
5 corresponding to a regular timbre and that determines a second system
6 combining a small number of operation blocks and corresponding to a
7 substitute timbre, and a changing device operative when a number of operation
8 blocks executable in the channel is limited under said great number and over
9 said small number due to a load of the computation of the waveform for
10 changing the musical tone from the regular timbre to the substitute timbre so
11 that the second system is adopted for the channel in place of the first system.

12 Preferably, the setting device comprises an adjusting device operative
13 dependently on a condition during the course of generating the musical tone for
14 adjusting a number of the operation blocks to be allocated to the channel.

15 Preferably, the adjusting device comprises a modifying device that
16 modifies the algorithm to eliminate a predetermined one of the operation blocks
17 involved in the system so as to reduce a number of the operation blocks to be
18 loaded into the channel for adjustment to the condition.

19 Preferably, the adjusting device operates when the condition indicates
20 that an amplitude envelope of the waveform attenuates below a predetermined
21 threshold level for compacting the system so as to reduce the number of the
22 operation blocks.

23 Preferably, the adjusting device operates when the condition indicates
24 that an output volume of the musical tone is tuned below a predetermined
25 threshold level for compacting the system so as to reduce the number of the
26 operation blocks.

1 Preferably, the adjusting device operates when the condition indicates
2 that one of the operation blocks declines to become inactive in the system
3 without substantially affecting other operation blocks of the system for
4 eliminating said one operation block so as to reduce the number of the
5 operation blocks to be allocated to the channel.

6 Preferably, the generating device comprises a computing device
7 responsive to a variable sampling frequency for executing the operation blocks
8 to successively compute samples of the waveform in synchronization to the
9 variable sampling frequency so as to generate the musical tone, and a
10 controlling device that sets the variable sampling frequency according to
11 process of computation of the waveform by the operation blocks.

12 Preferably, the generating device comprises a computing device
13 responsive to a variable sampling frequency for executing the operation blocks
14 to successively compute samples of the waveform in synchronization to the
15 variable sampling frequency so as to generate the musical tone, and a
16 controlling device for adjusting the variable sampling frequency dependently
17 on a load of computation of the waveform during the course of generating the
18 musical tone .

19 Preferably, the generating device comprises a computing device
20 responsive to a variable sampling frequency for executing the operation blocks
21 to successively compute samples of the waveform in synchronization to the
22 variable sampling frequency so as to generate the musical tone, and a
23 controlling device for adjusting the variable sampling frequency according to
24 result of computation of the samples during the course of generating the
25 musical tone.

0822947-0822997

1 The inventive sound source apparatus has a software module used to
2 compute samples of a waveform in response to a sampling frequency for
3 generating a musical tone according to performance information. In the
4 inventive apparatus, a processor periodically executes the software module for
5 successively computing samples of the waveform corresponding to a variable
6 sampling frequency so as to generate the musical tone. A detector device
7 detects a load of computation imposed on the processor device during the
8 course of generating the musical tone. A controller device operates according
9 to the detected load for changing the variable sampling frequency to adjust a
10 rate of computation of the samples.

11 Preferably, the controller device provides a fast sampling frequency
12 when the detected load is relatively light, and provides a slow sampling
13 frequency when the detected load is relatively heavy such that the rate of the
14 computation of the samples is reduced by $1/n$ where n denotes an integer
15 number.

16 Preferably, the processor device includes a delay device having a
17 memory for imparting a delay to the waveform to determine a pitch of the
18 musical tone according to the performance information. The delay device
19 generates a write pointer for successively writing the samples into addresses of
20 the memory and a read pointer for successively reading the samples from
21 addresses of the memory to thereby create the delay corresponding to an address
22 gap between the write pointer and the read pointer..The delay device is
23 responsive to the fast sampling frequency to increment both of the write pointer
24 and the read pointer by one address for one sample. Otherwise, the delay
25 device is responsive to the slow sampling frequency to increment the write

1 pointer by one address n times for one sample and to increment the read pointer
2 by n addresses for one sample.

3 Preferably, the processor device includes a delay device having a pair of
4 memory regions for imparting a delay to the waveform to determine a pitch of
5 the musical tone according to the performance information. The delay device
6 successively writes the samples of the waveform of one musical tone into
7 addresses of one of the memory regions, and successively reads the samples
8 from addresses of the same memory region to thereby create the delay. The
9 delay device is operative when said one musical tone is switched to another
10 musical tone for successively writing the samples of the waveform of said
11 another musical tone into addresses of the other memory region and
12 successively reading the samples from addresses of the same memory region to
13 thereby create the delay while clearing the one memory region to prepare for a
14 further musical tone.

15 Preferably, the processor device executes the software module composed
16 of a plurality sub-modules for successively computing the waveform. The
17 processor device is operative when one of the sub-modules declines to become
18 inactive without substantially affecting other sub-modules during computation
19 of the waveform for skipping execution of said one sub-module.

20 The inventive sound source apparatus has a software module used to
21 compute samples of a waveform for generating a musical tone. In the inventive
22 apparatus, a provider device variably provides a trigger signal at a relatively
23 slow rate to define a frame period between successive trigger signals, and
24 periodically provides a sampling signal at a relatively fast rate such that a
25 plurality of sampling signals occur within one frame period. A processor
26 device is resettable in response to each trigger signal and is operable based on

1 each sampling signal to periodically execute the software module for
2 successively computing a number of samples of the waveform within one frame.
3 A detector device detects a load of computation imposed on the processor
4 device during the course of generating the musical tone. A controller device is
5 operative according to the detected load for varying the frame period to adjust
6 the number of the samples computed within one frame period. A converter
7 device is responsive to each sampling signal for converting each of the samples
8 into a corresponding analog signal to thereby generate the musical tones.

9 10 **BRIEF DESCRIPTION OF THE DRAWINGS**

11 The above and other objects, features and advantages of the present
12 invention will become more apparent from the accompanying drawings, in
13 which like reference numerals are used to identify the same or similar parts in
14 several views.

15 \ FIG. 1 is a schematic block diagram illustrating a software constitution
16 of a sound source system practiced as a first preferred embodiment of the
17 present invention;

18 \ FIG. 2 is a block diagram illustrating a general hardware constitution of
19 the sound source system practiced as the first preferred embodiment of the
20 present invention;

21 \ FIG. 3 is a diagram for explaining music tone generation processing
22 performed by the sound source system of FIG. 1;

23 \ FIGS. 4A through 4C are a diagram for explaining overview of the music
24 tone generation processing based on an FM sound source;

25 \ FIG. 5 is a diagram illustrating examples of basic waveform data selected
26 from a basic waveform table;

1 \ FIG. 6 is a diagram illustrating a timbre register used for expanding
2 timbre parameters of a music tone to be sounded through an assigned channel;

3 \ FIGS. 7A through 7C are a diagram illustrating a data format of music
4 tone parameter VOICEj;

5 \ FIG. 8 is a diagram illustrating a MIDI-CH voice table for storing a voice
6 number of music tone parameter VOICE_n selectively set in each MIDI channel;

7 \ FIG. 9 is a flowchart indicating procedure of an initialization program
8 executed by the CPU of the sound source system of FIG. 1;

9 \ FIG. 10 is a flowchart indicating procedure of a main program executed
10 by the CPU after the initialization program of FIG. 9;

11 \ FIG. 11 is a flowchart indicating detailed procedure of a MIDI
12 processing subroutine contained in the main routine of FIG. 10;

13 \ FIG. 12 is a flowchart indicating a continued part from the MIDI
14 processing subroutine of FIG. 11;

15 \ FIG. 13 is a diagram illustrating an example of a format of a CH
16 sequence register;

17 \ FIG. 14 is a flowchart indicating detailed procedure of a waveform
18 computation processing subroutine contained in the main routine of FIG. 10;

19 \ FIG. 15 is a flowchart indicating a continued part from the waveform
20 computation processing subroutine of FIG. 14;

21 \ FIG. 16 is a flowchart indicating detailed procedure of an FM
22 computation processing subroutine for one channel;

23 \ FIG. 17 is a flowchart indicating detailed procedure of an operator
24 computation processing subroutine for one operator;

25 \ FIG. 18 is a flowchart indicating a continued part from the operator
26 computation processing subroutine;

1 \ FIG. 19 is a diagram illustrating a basic flow of an operator computation
2 performed in the operator computation processing of FIGS. 17 and 18;

3 \ FIG. 20 is a flowchart indicating detailed procedure of a timbre setting
4 processing subroutine contained in the main routine of FIG. 10;

5 \ FIG. 21 is a diagram illustrating a constitution of a software sound
6 source system practiced as a second preferred embodiment of the present
7 invention;

8 \ FIG. 22 is a diagram illustrating an operation timing chart of the software
9 sound source system shown in FIG. 21;

10 \ FIG. 23 is a block diagram illustrating a processing apparatus having a
11 tone waveform data generator implemented according to the tone waveform
12 data generating method of the present invention;

13 \ FIG. 24 is a block diagram illustrating a constitutional example of a
14 tube/string model section of a sound source model implemented according to
15 the tone waveform data generating method of the present invention;

16 \ FIG. 25 is a block diagram illustrating a constitutional example of a
17 timbre effect attaching section provided in the sound source model
18 implemented according to the tone waveform data generating method of the
19 present invention;

20 \ FIG. 26 is a block diagram illustrating a constitutional example of an
21 exciter section provided in the sound source model implemented according to
22 the tone waveform data generating method of the present invention;

23 \ FIG. 27 is a diagram illustrating a variety of data expanded in a RAM
24 shown in FIG. 23;

1 \ FIG. 28 is a diagram illustrating details of control parameter
2 VATONPAR necessary for computational generation of musical tones in the
3 present invention;

4 \ FIG. 29 is a flowchart of an initialization program used in the present
5 invention;

6 \ FIG. 30 is a flowchart of a main program in the present invention;

7 \ FIG. 31 is a flowchart of MIDI processing in the main program;

8 \ FIGS. 32A through 32C are a flowchart of physical model sound source
9 key-on processing in the MIDI processing, a flowchart of other MIDI event
10 processing and a flowchart of timbre setting processing activated by a user;

11 \ FIG. 33 is a flowchart of physical model parameter expansion processing
12 in the timbre setting processing;

13 \ FIG. 34 is a part of a flowchart of waveform generation processing of a
14 physical model sound source of the present invention;

15 \ FIG. 35 is the remaining part of the flowchart of the waveform
16 generation processing of the physical model sound source of the present
17 invention;

18 \ FIG. 36 is a flowchart of physical model computation processing in the
19 tone waveform generation;

20 \ FIG. 37 is a flowchart of delay loop section computation processing in
21 the physical model sound source computation processing;

22 \ FIG. 38 is a diagram for explaining a method of controlling a delay time
23 length of a delay circuit of the physical model sound source;

24 \ FIG. 39 is a diagram for explaining a method of controlling a delay time
25 length in the physical model sound source;

1 FIGS. 40A and 40B are a diagram illustrating a storage state of the
2 control parameter VATONEPAR of each timbre;

3 FIG. 41 is a diagram illustrating a hardware constitution of a delay
4 circuit in the physical model sound source associated with the present
5 invention;

6 FIGS. 42A and 42B are a diagram for explaining an operation mode of
7 the delay circuit shown in FIG. 41;

8 FIG. 43 is a diagram for explaining allocation of a delay memory area in
9 a delay circuit included in the physical model sound source associated with the
10 present invention; and

11 FIG. 44 is a diagram for explaining allocation of a delay circuit in a
12 physical model sound source having a plurality of sound channels.

14 **DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS**

15 This invention will be described in further detail by way of example with
16 reference to the accompanying drawings. FIG. 1 shows a software constitution
17 of a sound source system practiced as a first preferred embodiment of the
18 present invention. As shown in the figure, this software sound source system is
19 constituted to generate music tone waveform data based on an operating system
20 (OS). It should be noted that FIG. 1 also shows CODEC hardware including a
21 DAC (Digital to Analog Converter) for converting a digital music signal in the
22 form of the music tone waveform data generated under control of the OS into
23 an analog music tone signal.

24 Now, referring to FIG. 1, an APS1 is application software such as a
25 sequencer software operable on real-time basis for sequentially generating
26 performance information containing MIDI messages. The sequencer software

1 APS1 has a plurality of MIDI files composed of MIDI data such as various
2 event data and timing data for timing occurrence of the event data. The MIDI
3 file is prepared for generating pieces of music. When one or more MIDI files
4 are selected by the user, the MIDI data is read sequentially from the selected
5 files. Based on the read MIDI data, MIDI messages are sequentially generated
6 according to the event data at real-time. Then, the sequencer software APS1
7 outputs the generated MIDI messages to a first interface IF1 which is a MIDI
8 Application Interface or MIDI API arranged on the OS for MIDI message
9 input.

10 The OS is installed with a driver defining a software sound source
11 module SSM. This module is a program for generating music tone waveform
12 data based on the MIDI messages inputted via the first interface IF1. The OS
13 also has a second interface IF2 denoted by WAVE out Application Interface or
14 WAVE out API for receiving the music tone waveform data generated by the
15 software sound source module SSM. Further, the OS is installed with an output
16 device OUD which is a software driver for outputting the music tone waveform
17 data inputted via the second interface IF2. To be more specific, this output
18 device OUD reads, via a direct memory access (DMA) controller, the music
19 waveform data generated by the software sound source module SSM and
20 temporarily stored in a storage device such as a hard disk, and outputs the read
21 music waveform data to a predetermined hardware device such as a CODEC.

22 The MIDI messages outputted by the sequencer software APS1 are
23 supplied to an input interface of the software sound source module SSM via the
24 first interface IF1 and the OS. The software sound source module SSM
25 performs music tone waveform data generation processing. In the present
26 embodiment, the music tone waveform data is generated by FM tone generating

1 based on the received MIDI messages. The generated music tone waveform
2 data is supplied to the output device OUD via the second interface IF2 and the
3 OS. In the output device OUD, the supplied music tone waveform data is
4 outputted to the above-mentioned CODEC to be converted into an analog
5 music tone signal.

6 Thus, the present embodiment allows, at the OS level, ready combination
7 of the software sound source module SSM for generation music tone waveform
8 data and the sequencer software APS1 which is the application software for
9 outputting MIDI messages. This makes it unnecessary to add any hardware
10 components dedicated to music tone waveform data generation, resulting in
11 reduced cost.

12 FIG. 2 shows an overall hardware constitution for implementing the
13 sound source system of the present embodiment. This system is implemented
14 by a general-purpose personal computer. For the main controller of this system,
15 a CPU 1 is used. Under the control of the CPU 1, the music tone waveform data
16 generation processing by a software sound source program and processing by
17 other programs are executed in parallel under multi-tasks.

18 Referring to FIG. 2, the CPU 1 is connected, via a data/address bus 19, to
19 a MIDI interface (MIDI I/F) 12 for inputting MIDI messages from an external
20 device and for outputting MIDI messages to an external device, a timer 16 for
21 counting a timer interrupt time and other various times, a ROM (Read Only
22 Memory) 2 for storing various control programs and table data, a RAM
23 (Random Access Memory) 3 for temporarily storing a selected MIDI file,
24 various input information, and computational results, a mouse 17 used as a
25 pointing device, an alphanumeric keyboard 10 through which character
26 information is mainly inputted, a display 5 composed of a large-sized LCD or a

1 music tone generation processing based on the MIDI messages supplied within
2 a preceding frame from time t1 to time t2 is executed in a succeeding frame
3 from time t2 to time t3.

4 When the music tone waveform data for one frame has been generated by
5 the music tone generation processing, the generated music tone waveform data
6 is written to the output buffer of the RAM 3. Reproduction of the written data
7 is reserved in the output device OUD. This reservation in the OUD is
8 equivalent to the outputting of the generated music tone waveform data from
9 the software sound source module SSM to the second interface IF2 (WAVE out
10 API) of the OS level.

11 The output device OUD reads the music tone waveform data, a sample
12 by sample, from the output buffer reserved for the reproduction in the
13 immediately preceding frame, and outputs the data to the DAC 14b. For
14 example, as shown in FIG. 3, the music tone waveform data generated in the
15 frame from time t2 to time t3 and written to the output buffer for reserved
16 reproduction is read in a next frame from time t3 to time t4 for the sound
17 reproduction.

18 The following is an overview of the music tone generation processing
19 based on music tone parameter VOICE_n. In this embodiment, the music tone
20 generation processing is based on FM tone generating as shown in FIGS. 4A
21 through 4C. FIG. 4A through FIG. 4C show three different music tone
22 generating methods. As shown in the figures, the music tone generation based
23 on FM tone generating is performed by combining two types of operation
24 blocks or operators, namely, an operator called a carrier and an operator called
25 modulator. The different number of combined operators and the different
26 connection sequences (connection modes) are used according to the type and

1 quality of the music tone waveform to be generated. Systematic connection
2 scheme of these operators is called an algorithm.

3 The operator herein denotes a block that provides a unit in which tone
4 creation or music tone generation processing is performed. To be more specific,
5 from various basic waveform data used for the tone creation, one piece of basic
6 waveforms shown in FIG. 5 for example is selected according to a wave select
7 parameter WSEL and is read based on input data such as pitch data and
8 modulation data. If the input data includes two types of data such as the pitch
9 data and the modulation data, the basic waveform data is read out based on a
10 result obtained by adding these two pieces of data together. Then, the
11 amplitude of this one piece of waveform data is adjusted, and the adjusted data
12 is outputted. The operation block in which these operations are performed is
13 called the operator. Among the operators, the carrier denotes an operator for
14 generating a basic music tone waveform. The modulator denotes an operator
15 for modulating the carrier, namely for generating modulation data for
16 modulating the carrier. It should be noted that the algorithm is not limited to
17 the three types shown in FIGS. 4A through 4C.

18 The following explains a data format of the above-mentioned music tone
19 parameter VOICEj. FIGS. 7A through 7C show the data format of the music
20 tone parameter VOICEj. FIG. 7A shows the data format of the music tone
21 parameter VOICEj, FIG. 7B shows a data format of each operator data
22 OPmDATAj shown in FIG. 7A, and FIG. 7C shows a data format of each
23 operator buffer OPBUFm shown in FIG. 7B.

24 As shown in FIG. 7A, the music tone parameter VOICEj is composed of
25 key-on data KEYONj indicating key-on and key-off by "1" and "0"
26 respectively, frequency number FNOj (actually represented by a phase rate)

1 determined by pitch information included in a MIDI message of a
2 corresponding note-on event, algorithm designation data ALGORj for
3 designating one of the above-mentioned algorithms, volume data VOLj
4 determined according to volume set to a MIDI channel concerned. The volume
5 is set by control change #7 event of the MIDI message, for example. The
6 music tone parameter further contains touch velocity data VELj determined
7 according to touch velocity information in the MIDI message concerned, and
8 operator data OPkDATAj (j = 1 to m) made up of a buffer for holding data
9 necessary for computing music tone generation in each of the constituent
10 operators and the results of this computation.

11 It should be noted that the music tone parameter VOICEj simultaneously
12 has two types of data, one type read from the ROM 2, RAM 3, or the hard disk
13 and the other type determined according to the data in the MIDI message. The
14 data determined according to the MIDI message includes the key-on data
15 KEYONj, the frequency number FNOj, the volume data VOLj, and the touch
16 velocity data VELj. The data read from the ROM 2 and so on includes the
17 algorithm designation data ALGORj and the operator data OPkDATAj.

18 As shown in FIG. 7B, each operator data OPkDATAj is composed of
19 sampling frequency designation data FSAMPm for designating a sampling
20 frequency used in operator m, frequency multiple data MULTm providing a
21 parameter for substantially setting a frequency ratio between operators
22 (actually, a parameter for designating an integer multiple for varying the
23 above-mentioned frequency number FNOj), feedback level data FBLm
24 indicating a feedback level (namely, a degree of feedback modulation), wave
25 select data WSELM for selecting basic waveform data to be used by operator m
26 from various pieces of basic waveform data (described with reference to FIG.

1 5) stored in the ROM 2, total level data TL_m for setting the output level
2 (varying with the above-mentioned touch velocity data VEL_j) of a music tone
3 waveform to be generated in the operator m , envelope parameter $EGPAR_m$
4 composed of various type of data (for example, attack time, decay time, sustain
5 level, and release time) for determining the envelope of a music tone waveform
6 to be generated in the operator m , data MSC_m indicating other parameters (for
7 example, velocity and depth of vibrato and tremolo, and various key scaling
8 coefficients), operator priority data $OPPRIO_m$ indicating priority of operator m
9 (for example, priorities of start and stop of the waveform generating
10 computation in each operator), and buffer $OPBUF_m$ for storing the results of
11 the music tone waveform generating computation in operator m .

12 The sampling frequency designation data $FSAMP_m$ contains integer
13 value f higher than "0". This integer value f allows the sampling frequency
14 $FSMAX$ (for example, 44.1 kHz) in standard mode to be multiplied by 2^f . For
15 example, if $f = 0$, a music tone waveform in operator m is generated at the
16 sampling frequency $FSMAX$ of the standard mode; if $f = 1$, a music tone
17 waveform in operator m is generated at the sampling frequency of $FSMAX/2$.

18 The operator priority data $OPPRIO_m$ contains data (for example,
19 numbers indicating the order by which waveform computing operations are
20 performed) indicating the priority of the waveform computation processing in
21 all operators k ($k = 1$ to m). According to this priority data, the priority by
22 which each operator is activated is determined for the waveform computation
23 processing. Alternatively, the performance and load states of the CPU 1 are
24 checked to determine the operators to be activated. If this check indicates that
25 the CPU 1 has no more capacity for performing tone generation processing, the
26 computation processing of the operators of lower priorities may be left out. In

the present embodiment, the priorities of the computation processing are set according to timbre applied to the music tone. Alternatively, the priorities may be set according to MIDI channels for example. Namely, the priorities set by some reference may be selected for use at sounding. For example, if the priorities are not set according to the timbre, the operator priority data OPPRIOM may be determined based on the timbre parameter expanded in the above-mentioned timbre register TONEPARn. The operator priority data OPPRIOM may be handled also as to determine the setting that operator m is to be used or not.

In the present embodiment, the sampling frequency can be set for each operator m by the above-mentioned sampling frequency designation data FSAMPm. Alternatively, the sampling frequency may be set differently for the two types of the operators, the carrier and the modulator. For example, the carrier may be set to the above-mentioned frequency FSMAX and the modulator may be set to 1/2 of the FSMAX. In this case, the contents of the algorithm of the timbre parameter concerned are checked and the sampling frequency may be accordingly set for the operators with which the timbre parameter is combined. Alternatively, the load state of the CPU 1 is checked and the sampling frequency may be accordingly increased or decreased.

As shown in FIG. 7C, the buffer OPBUFm is composed of operator-on parameter OPONm indicating by "1" that the waveform computation is performed by operator m (namely, operator m is on), phase value buffer PHBUFm for storing a phase value obtained by performing phase computation on the result of the waveform computation performed by operator m, feedback output value buffer FBm for storing a feedback output value obtained by the feedback sample computation of the above-mentioned waveform computation

1 processing, modulation data input buffer MODINm for storing modulation data
2 (this data is used in the above-mentioned phase computation processing),
3 operator output value buffer OPOUTm for storing the music tone waveform
4 (namely the output value) generated by operator m, and EG state buffer
5 EGSTATEm for storing the EG parameters obtained by the computation
6 processing (hereafter referred to as AEG computation processing) for
7 computing amplitude controlling EG of the above-mentioned waveform
8 computation processing.

9 FIG. 8 shows a MIDI-CH voice table for storing voice data
10 representative of a timbre selectively set for each MIDI channel or for each
11 performance part of the music piece. In the present embodiment, the voice data
12 is denoted by a voice number of music tone parameter VOICE_n.

13 As shown in FIG. 8, in the present embodiment, 16 MIDI channels are
14 provided. Different timbres can be set to different MIDI channels
15 corresponding to different performance parts. Consequently, the sound source
16 system of the present embodiment can generate a maximum of 16 types of
17 timbres. This MIDI-CH voice table lists the voice numbers of the timbres
18 assigned to the sound channels, namely the voice numbers contained in the
19 above-mentioned music tone parameters VOICE_n.

20 The MIDI-CH voice table is allocated at a predetermined area in the
21 RAM 3. The table data, namely the voice numbers, are stored beforehand on
22 the hard disk or the like in correspondence with the selected MIDI file. The
23 user-selected MIDI file is loaded into a performance data storage area allocated
24 at a predetermined location in the RAM 3. At the same time, the table data
25 corresponding to the loaded MIDI file is loaded into the MIDI-CH voice table.
26 Alternatively, the user can arbitrarily set the MIDI-CH voice table from the

1 beginning or can change the table after standard voice numbers have been set to
2 the music piece. MIDI messages are sequentially generated by the sequencer
3 program APS1 and the generated MIDI messages are recognized by the
4 software sound source module SSM. The software sound source module SSM
5 then searches the MIDI-CH voice table for the voice number assigned to the
6 MIDI channel of the MIDI message concerned. For example, if the MIDI
7 channel of the MIDI message concerned is "2HC," the voice number stored at
8 the second location VOICENO2 in the MIDI-CH voice table is selected.

9 When voice number j is found, the software sound source module SSM
10 generates music tone parameter VOICE j as described above. To be more
11 specific, the software sound source module SSM reads the basic data from the
12 ROM 2 and determines other parameters from the MIDI message concerned to
13 generate the music tone parameter VOICE j shown in FIGS. 7A through 7C.
14 Then, the software sound source module SSM expands the generated music
15 tone parameter VOICE j in a timbre register TONEPAR n corresponding to the
16 sound channel among the plurality of timbre registers shown in FIG. 6.

17 As described above, the inventive sound source apparatus has the
18 operation blocks OPs (shown in FIGS. 4A through 4C) composed of softwares
19 used to compute waveforms for generating a plurality of musical tones through
20 a plurality of sound channels according to performance information in the form
21 of the MIDI messages. In the inventive apparatus, a setting device sets an
22 algorithm (shown in FIGS. 4A through 4C) which determines a system of the
23 software sound source module SSM composed of selective ones of the
24 operation blocks OPs systematically combined with each other to compute a
25 waveform specific to one of the musical tones. A designating device including
26 the MIDI API shown in FIG. 1 responds to the performance information for

08920947-082997

1 designating one of the channels to be used for generating said one musical tone.
2 A generating device including the CPU 1 allocates the selective operation
3 blocks to said one channel and systematically executes the allocated selective
4 operation blocks according to the algorithm so as to compute the waveform to
5 thereby generate said one musical tone through said one channel.

6 Preferably, the setting device sets different algorithms which determine
7 different systems corresponding to different timbres of the musical tones. Each
8 of the different systems is composed of selective ones of the operation blocks
9 which are selectively and sequentially combined with each other to compute a
10 waveform which is specific to a corresponding one of the different timbres.

11 Preferably, the setting device comprises a determining device that
12 determines a first system combining a great number of operation blocks and
13 corresponding to a regular timbre and that determines a second system
14 combining a small number of operation blocks and corresponding to a
15 substitute timbre, and a changing device operative when a number of operation
16 blocks executable in the channel is limited under said great number and over
17 said small number due to a load of the computation of the waveform for
18 changing the musical tone from the regular timbre to the substitute timbre so
19 that the second system is adopted for the channel in place of the first system.

20 Preferably, the setting device comprises an adjusting device operative
21 dependently on a condition during the course of generating the musical tone for
22 adjusting a number of the operation blocks to be allocated to the channel.

23 Preferably, the adjusting device comprises a modifying device that
24 modifies the algorithm to eliminate a predetermined one of the operation blocks
25 involved in the system so as to reduce a number of the operation blocks to be
26 loaded into the channel for adjustment to the condition.

27

1 Preferably, the generating device comprises a computing device
2 responsive to a variable sampling frequency for executing the operation blocks
3 to successively compute samples of the waveform in synchronization to the
4 variable sampling frequency so as to generate the musical tone, and a
5 controlling device for adjusting the variable sampling frequency according to
6 result of computation of the samples during the course of generating the
7 musical tone.

8 The following explains the control processing to be performed by the
9 sound source system thus constituted, with reference to FIGS. 9 through 20.
10 FIG. 9 is a flowchart showing the procedure of an initialization program to be
11 executed by the CPU 1 in the sound source system of the present embodiment.
12 The initialization program is executed when the user turns on the power to the
13 sound source system, or presses a reset switch thereof. First, system
14 initialization such as resetting ports and clearing the RAM 3 and a video RAM
15 in the display 5 is performed (step S1). Next, the OS program is read from the
16 hard disk of the hard disk drive 6 for example, and the OS program is loaded in
17 a predetermined area in the RAM 3 so as to run the OS program (step S2). Then,
18 the process goes to the execution of a main program.

19 FIG. 10 is a flowchart indicating the procedure of the main program to be
20 executed by the CPU 1 after execution of the initialization program. This main
21 program is the main routine of the software sound source module SSM. First,
22 the area containing the timbre register group shown in FIG. 6 in the RAM 3 to
23 be used by the software sound source module SSM is cleared. At the same time,
24 the various types of basic data (for example, the various pieces of basic
25 waveform data shown in FIG. 5) stored in the hard disk of the hard disk drive 6
26 are loaded in a predetermined area in the RAM 3 (step S11). Next, basic

1 SSM passes control to step S19, in which a timbre setting processing
2 subroutine is executed especially if a timbre setting event has occurred; if
3 another event has occurred, corresponding processing is performed in step S20.
4 If the trigger 5 has taken place, the software sound source module SSM passes
5 control to step S21, in which end processing such as returning the screen of the
6 display 5 to the initial state provided before the main program was started.
7 Then, any of the steps S16 through S21 has been ended, the software sound
8 source module SSM passes control to step S12 to repeat the above-mentioned
9 operations.

10 FIGS. 11 and 12 are flowcharts indicating the detailed procedure of the
11 MIDI processing subroutine of step S16. First, the software sound source
12 module SSM checks to see whether a MIDI event (a MIDI message) has been
13 inputted via the software sound source interface API of the software sound
14 source module SSM (step S31). When a MIDI message is outputted from the
15 sequencer software APS1, the outputted MIDI message is converted in a
16 predetermined manner by the first interface IF1 and the OS. The converted
17 MIDI message is then transferred to a MIDI event buffer allocated at a
18 predetermined area in the RAM 3 via the software sound source interface API.
19 When this transfer is made, the software sound source module SSM determines
20 that the trigger 1 has taken place, thereby passing control by the CPU 1 from
21 step S15 to step S16. The processing operations so far are performed in the
22 preparation processing of step S20 in the main routine of FIG. 10. In step S31,
23 the software sound source module SSM monitors the event occurrence by
24 checking the MIDI event buffer.

25 Next, in step S32, the software sound source module SSM determines
26 whether the MIDI event is a note-on event. If the MIDI event is found a note-

1 on event, the software sound source module SSM passes control to step S33; if
2 not, the SSM passes control to step S40 shown in FIG. 12. In step S33, the
3 SSM decodes the note-on event data and stores resultant note-number data,
4 velocity value data and part number data (namely, the MIDI channel number)
5 into registers NN, VEL, and p, respectively. Further, the SSM stores the data
6 about the time at which the note-on event should take place into a register TM
7 allocated at a predetermined position in the RAM 3. Hereafter, the contents of
8 the registers NN, VEL, p, and TM are referred to as note number NN, velocity
9 VEL, part p, and time TM, respectively.

10 In step S34, the software sound source module SSM determines whether
11 velocity VEL is lower than a predetermined value VEL1 and whether volume
12 data VOLp is lower than a predetermined value VOL1. The VOLp denotes the
13 volume data of the part p stored in area VOLp allocated at a predetermined area
14 in the RAM 3. This VOLp is changed by the control change #7 event of the
15 MIDI message as explained with reference to FIG. 7A. The change is
16 performed in the miscellaneous processing of step S20 when the control change
17 #7 event has taken place. In step S34, if $VEL \leq VEL1$ and $VOL \leq VOL1$, the
18 regular timbre allotted to the part p is replaced by a substitute timbre of an
19 algorithm having a small number of operators, namely a small total number of
20 carriers and modulators. That is, the voice number stored in VOICEp of the
21 part p in the above-mentioned MIDI-CH voice table is replaced by the voice
22 number of the music tone parameter VOICE having an alternate algorithm (step
23 S35). If $VEL > VEL1$ or $VOL > VOL1$, the SSM skips step S35 and passes
24 control to step S36. In the present embodiment, whether the processing of step
25 S35 is to be performed is determined according to the values of velocity VEL

1 and volume VOL. The decision may also be made by detecting the load state of
2 the CPU 1 and according to the detection result, for example.

3 In step S36, channel assignment processing based on the note-on event
4 concerned is performed. The channel number of the assigned sound channel is
5 stored in register n allocated at a predetermined location in RAM 3. The
6 contents stored in the register n are hereafter referred to as sound channel n. In
7 step S37, the MIDI-CH voice table shown in FIG. 8 is searched. The timbre
8 data (voice number) of VOICENOp of the part p in the table is converted into a
9 music tone parameter according to the above-mentioned note number NN and
10 velocity VEL. For example, if voice number j is stored in VOICENOp, the
11 music tone parameter VOICEj explained with reference to FIG. 7A is generated.
12 Then, the buffer OPBUFm in each operator data OPmDATAj of the music tone
13 parameter VOICEm is initialized or cleared.

14 In step S38, the music tone parameter VOICEj generated in step S37 is
15 transferred or expanded along with time TM into the timbre register
16 TONEPARn corresponding to the sound channel n. At the same time, key-on
17 data KEYONn in the timbre register TONEPARn and each operator-on
18 parameter OPONm are set to "1" (on). Further, in step S39, the computational
19 order is determined among the sound channels assigned for sounding such that
20 music tone generating computations are performed in the order of note-on
21 event occurrence times. To be more specific, the channel numbers are
22 rearranged according to the determined computational order and the rearranged
23 channel numbers are stored in CH sequence register CHSEQ allocated at a
24 predetermined position in the RAM 3, upon which this MIDI processing comes
25 to an end. The CH sequence register CHSEQ is illustrated in FIG. 13.

1 In step S40 of FIG. 12, it is determined whether the MIDI event is a
2 note-off event. If the MIDI event is found a note-off event, the SSM passes
3 control to step S41; otherwise, the SSM passes control to step S44. In step S41,
4 the note-off event data concerned is decoded. The note number turned off is
5 stored in the register NN. At the same time, data indicating the time at which
6 the note-off event should occur is stored in the register TM. In step S42, the
7 sound channel with the note number NN assigned for sounding is searched.
8 The channel number obtained is stored in register i (this value is hereafter
9 referred to as "sound channel i") allocated at a predetermined position in the
10 RAM 3. In step S43, key-off is designated for timbre register TONEPARI
11 corresponding to sound channel i. Namely, after note-off is reserved in the
12 timing corresponding to time TM, this MIDI processing is ended.

13 In step S44, it is determined whether the MIDI event is a program change
14 event for changing timbres. If the MIDI event is found a program change event,
15 the data of VOICENOp at the position corresponding to the part p (this part p is
16 not necessarily the part number stored in step S33) designated by the received
17 program change event is changed to value PCHNG designated by the received
18 program change event, upon which this MIDI processing comes to an end (step
19 S45). On the other hand, if the MIDI event is found other than a program
20 change event, the corresponding processing is performed, upon which this
21 MIDI processing comes to an end.

22 In this MIDI processing, the timbres corresponding to a plurality of parts
23 are designated in the MIDI-CH voice table. If a note-on event of a plurality of
24 designated parts occurs, a music tone having timbres of the plurality of parts is
25 generated and sounded. Namely, this MIDI processing uses multi-timbre
26 operation specifications. Alternatively, this MIDI processing may use a

1 single-timbre mode in which only a note-on event of a particular part is
2 accepted to generate a music tone of the corresponding timbre.

3 FIGS. 14 and 15 are flowcharts indicating detailed procedures of the
4 waveform computation processing subroutine performed in step S17 of FIG. 10.
5 First, a music tone waveform buffer is initialized (step S51). A music tone
6 waveform buffer exists in an area other than a reserved area (buffer) for
7 reproduction in the output buffer. The music tone waveform buffer provides an
8 area for one frame time of waveforms to be generated this time. The
9 initialization of this music tone waveform buffer is to allocate that area in the
10 output buffer and to clear that area. Next, the load state of the CPU 1 is checked
11 (step S52). Based on the check result, a maximum number of channels CHmax
12 that can execute the waveform computation processing is determined (step
13 S53). If the OS always checks the load state of the CPU 1, the check of step
14 S52 may be performed using this load state information. If the OS does not
15 always check the load state of the CPU 1, a routine may be provided that counts
16 a time for looping the main program of FIG. 10 once. The check of step S52
17 may be performed using a value obtained based on the measured time. Instead
18 of the processing of step S53, processing similar to the processing of step S35
19 of FIG. 11 may be performed. Namely, the timbre changing process is
20 conducted for changing the timbre assigned to the part to an alternate timbre
21 having a smaller number of constituting operators.

22 Then, index i indicating a channel number is initialized to "1"(step S54).
23 In step S55, the channel number SEQCHNO_i stored in SEQCH_i at i position in
24 the CH sequence register CHSEQ shown in FIG. 13 is stored in variable n (in
25 this waveform computation processing subroutine, this value is referred to as
26 "channel n"). In step S56, algorithm designation data ALGOR_n of the music

1 tone register TONEPARn corresponding to channel n is referenced to
2 determine the number of operators (OPs) and the connection mode of each
3 operator to be used in the FM computation processing for channel n.

4 Moreover, a computation amount in the current frame is determined
5 according to the note events and the like (step S57). The determination of the
6 computation amount actually denotes determining a net area of the music tone
7 waveform buffer for which the waveform computation processing is to be
8 performed in channel n. The music tone waveform buffer is the area sufficient
9 to store waveform data of one frame time in which the current computation is
10 made. On the other hand, the music tone waveform data of each channel is not
11 necessarily generated all over the area for one frame. Namely, since the
12 sounding timing and muting timing of music tones are different for different
13 channels, a music tone of a certain channel may be turned on or off halfway in
14 the music tone waveform buffer. In view of this, the computation amount must
15 be determined for each channel.

16 Next, in step S58 of FIG. 15, the FM computation processing subroutine
17 for generating music tone waveform data for one sample is generated for
18 channel n. In step S59, it is determined whether the music tone generation
19 processing for one frame for channel n has been completed. It should be noted
20 that the determination of step S59 is performed by considering the computation
21 amount determined in step S57. In step S59, if the music tone generation
22 processing for one frame for channel n has not been completed, the SSM passes
23 control back to step S58, in which the music tone waveform data of next sample
24 is generated. If, in step S59, the music tone generation processing for one
25 frame for channel n has been completed, the SSM passes control to step S60.

1 In step S60, the music tone waveform data for one frame generated in
2 steps S58 and S59 is written to the music waveform buffer. At this moment, if
3 music tone waveform data is already stored in the music waveform buffer, the
4 data obtained this time is accumulated to the existing data and a result of the
5 addition is written to the music tone waveform buffer. Then, the value of index
6 i is incremented by one (step S61) to determine whether the resultant value of
7 index i is greater than the above-mentioned maximum number of channels
8 CHmax (step S62).

9 In step S62, if $i \leq \text{CHmax}$, or if there are more channels to be processed
10 for the waveform generation, the SSM returns control to step S55, in which the
11 above-mentioned processing operations are repeated. If $i > \text{CHmax}$, or if there
12 is no channel to be processed, muting channel processing for gradually
13 decreasing the size of a volume envelope is performed for the sound channel
14 turned off this time (step S63). In step S64, the music tone waveform data thus
15 generated is removed from the music tone waveform buffer, and the removed
16 data is passed to the CODEC hardware which is an output device. Then,
17 reproduction of the data is instructed, upon which this waveform computation
18 processing comes to an end.

19 If the velocity value of channel n gets smaller than a predetermined value,
20 the FM computation for that channel n may not be performed. In order to
21 implement this operation, step S71 is provided after the above-mentioned step
22 S55 as shown in FIG. 14. In step S71, it is determined whether touch velocity
23 data VELn in the timbre register TONEPARn of channel n is higher than
24 predetermined value VELn1. If $\text{VELn} \geq \text{VELn1}$, the SSM passes control to
25 step S56; if $\text{VELn} < \text{VELn1}$, key-off is designated for channel n in the similar

1 step S86. In step S86, the operator computation processing subroutine for the
2 operator m is executed. In step S87, the value of variable m is incremented by
3 one. In step S88, if there are more operators to be processed, the SSM returns
4 control to step S82, in which the above-mentioned processing operations are
5 repeated. If there is no more operator to be processed, the FM computation
6 processing for channel n comes to an end.

7 In steps S82 and S83, the load state of the CPU 1 is checked to determine
8 whether the computation of the operator m is to be performed. Alternatively,
9 the computation for the operators having lower priority may not be performed
10 regardless of the load state of the CPU 1. This can increase the number of
11 sound channels when the capacity of the CPU 1 is not so high.

12 FIGS. 17 and 18 are flowcharts indicating the detailed procedure of the
13 operator computation processing subroutine for the operator m performed in
14 step S86. FIG. 19 is a diagram illustrating the basic flow of the operator
15 computation to be performed in this operator computation processing. The
16 following explains the operator computation processing for the operator m with
17 reference to FIGS. 17 through 19. Referring to FIG. 17, it is determined
18 whether the operator-on parameter OPONm in the operator data OPmDATAn
19 of the operator m is on ("1") (step S91). If OPONm is "0", or the operator m
20 does not require operator computation, this operator computation processing is
21 ended immediately. If OPONm is "1", or the operator m requires operator
22 computation, the SSM passes control to step S92.

23 In step S92, it is determined whether the sampling frequency designation
24 data FSAMPm in the operator data OPmDATAn is "0" or not. Namely, it is
25 determined whether a music tone waveform is to be generated at the sampling
26 frequency FSAMPm of standard mode. If FSAMPm = "0", it indicates the

1 standard mode in which each operator performs the music tone waveform
2 generation at the standard sampling frequency . Then, AEGm computation is
3 performed according to the setting value of the envelope parameter EGPARm
4 in the operator data OPmDATAn. The result of this computation is stored in
5 the EG state buffer EGSTATEm (step S93).

On the other hand, if $FSAMP_m \neq "0"$, for example, $FSAMP_m = f$, the sampling frequency $FSMAX$ of the standard mode is multiplied by 2^{-f} and the music tone waveform generation is performed at the resultant frequency. Namely, in step S94, a parameter of which rate varies (hereafter referred to as a variable-rate parameter) in the envelope parameters $EGPAR_m$ is multiplied by 2^f to perform the AEG computation. The result is stored in the EG state buffer $EGSTATE_m$. The rate of the variable-rate parameter is multiplied by 2^f before the envelope generating computation for the following reason. Namely, since the sampling frequency is reduced to $FSMAX \times 2^{-f}$, the time variation of the variable-rate parameter of the envelope parameter $EGPAR_m$ is made faster to perform the music tone waveform generation at the sampling frequency concerned. Subsequently, the generated waveform samples are written to 2^f continuous addresses of the buffer, thereby making adjustment such that the resultant music tone has the same pitch as that of the original music tone. Thus, step S93 or S94 performs the computation of envelope data AEG_m as shown in FIG. 19.

22 In step S95, the data AEGm obtained by the AEGm computation is
23 multiplied by the value of a total level parameter TLM in the operator data
24 OPmDATAn to compute an output level AMPm (= AEGm x TLM) of the
25 operator m as shown in FIG. 19. Then, the amplitude controlling envelope data
26 AEGm computed in step S93 or S94 and the output level AMPm of the operator

1 m computed in step S95 are checked independently (step S96). Based on the
2 check results, it is determined whether the data value AEGm and the data value
3 AMPm are lower than a predetermined time and a predetermined level,
4 respectively, thereby determining in turn whether the operator m is to be
5 operated or not (step S97). In other words, it is determined whether the music
6 tone waveform computation in the operator m may be ended or not. If the
7 decision is YES, the SSM passes control to step S98; if the decision is NO, the
8 SSM passes control to step S101 shown in FIG. 18.

9 In step S98, it is determined whether the operator m is a carrier. If the
10 operator m is found a carrier, the SSM passes control to step S99. In step S99,
11 the buffer OPBUF for the operator m and the modulator modulating only the
12 operator m are cleared, the waveform computation is stopped, and this operator
13 computation processing is ended. Thus, if the operator m is a carrier, not only
14 the waveform computation of the operator m but also the waveform
15 computation of the modulator modulating only the operator m is stopped. The
16 carrier is an operator that eventually outputs the music tone waveform data as
17 shown in FIGS. 4A through 4C. If there is no output from the carrier, or if the
18 SSM passes control from step S97 to S99 via S98, it may be assumed that
19 nothing is outputted from the modulator preceding the carrier. If that
20 modulator is modulating another carrier, the waveform computation of that
21 modulator cannot be stopped. On the other hand, if the operator m is found not
22 a carrier in step S98, or the operator is a modulator, only the buffer OPBUFm
23 of the operator m is cleared to stop the waveform computation (step S100),
24 upon which this operator computation processing comes to an end.

25 In step S101 shown in FIG. 18, algorithm designation data ALGORN is
26 checked. In step S102, it is determined whether the operator m is being

1 modulated from another operator. In step S102, if the operator m is found
2 being modulated from another operator, the operator output data stored in the
3 operator output value buffer OPOUTk in each operator data OPkDATA_n under
4 modulation are added together, and the result is stored in the data input buffer
5 MODIN_m of the operator m (step S103). On the other hand, if the operator m is
6 not being modulated by another operator, the SSM passes control to step S104,
7 skipping step S103. In step S104, it is determined whether the sampling
8 frequency designation data FSAMP_m in the operator data OPmDATA_n is "0".
9 If FSAMP_m = "0", the SSM passes control to step S105; if FSAMP_m ≠ "0", the
10 SSM passes control to step S110.

11 In step S105, a phase value update computation is performed. The
12 updated result is stored in the phase value buffer PHBUF_m (the contents
13 thereof hereafter being referred to as phase value PHBUF_m) in the operator
14 data OPmDATA_n of the operator m. The phase value update computation
15 denotes herein the computation enclosed by dashed line A in FIG. 19. To be
16 more specific, computation $\text{MODIN}_m + \text{FB}_m + \text{FNO}_n \times \text{MULT}_m + \text{PHBUF}_m$
17 is performed. MODIN_m and FB_m denote the values stored in the modulation
18 data input buffer MODIN_m in the operator data OPmDATA_n and the feedback
19 output value buffer FB_m, respectively. FNO_n denotes the frequency number
20 FNO_n in the music tone parameter VOICE_n. MULT_m denotes the frequency
21 multiple data MULT_m in the operator data OPmDATA_n. PHBUF_m denotes
22 the last value of the values stored in the phase value buffer PHBUF_m in the
23 operator data OPmDATA_n.

24 In step S106, a table address is computed based on the phase value
25 PHBUF_m computed in step S105. From the basic waveform (for example, a
26 waveform selected from among the above-mentioned eight types of basic

1 waveforms) data selected according to the wave select data WSELm of the
2 operator m, data WAVEm (PHBUFm) at the position pointed by this computed
3 address is read. It should be noted that basic waveform data is referred to
4 "basic waveform table." The data WAVEm (PHBUFm) is multiplied by the
5 output level AMPm computed in step S95. The result is stored in the operator
6 output value buffer OPOUTm (= WAVEm (PHBUFm) x AMPm) of the
7 operator m.

8 In step S107, feedback sample computation is performed by the
9 following relation, storing the result in the feedback output value buffer FBm
10 of the operator m.

11 0.5 x (FBm + OPOUTm x FBLm)

OPOUT_m denotes the waveform sample data generated in step S106. FBL_m denotes the feedback level data FBL_m of the parameter m to be computed. The feedback sample computation is performed to prevent parasitic exciter from occurring.

16 In step S108, it is determined, as with step S98, whether the operator m is
17 a carrier or not. If the operator m is found a modulator, this operator
18 computation processing is ended immediately. On the other hand, if the
19 operator m is found a carrier, the waveform sample data OPOUTm generated in
20 step S106 is multiplied by the volume data VOLn of the music tone parameter
21 VOICEn. The multiplication result ($= \text{OPOUTm} \times \text{VOLn}$) is added to the
22 position indicated by the pointer for pointing the write position of this time in
23 the corresponding waveform buffer. Further, the value of this pointer is
24 incremented by one (step S109), upon which this operator computation
25 processing comes to an end.

1 In step S110, phase value update computation is performed, and the
2 result is stored in the phase value buffer PHBUFm. This computation
3 processing in step S110 differs from the computation processing in step S105
4 only in the added processing indicated by block B in FIG. 19. Since FSAMPm
5 = f ($\neq 0$), the phase value must be shifted by f bits, or the value of the phase
6 value buffer PHBUFm must be multiplied by 2^f to change the read address of
7 the basic waveform table to that obtained by multiplying the sampling
8 frequency FS MAX by 2^f . Next, likewise step S106, waveform sample
9 generation is performed by the following relation, storing the result in the
10 operator output value buffer OPOUTm.

11
$$\text{WAVEm} (2^f \times \text{PHBUFm}) \times \text{AMPm}$$

12 Then, likewise step S107, a feedback sample computation is performed
13 (step S112).

14 In step S113, it is determined likewise step S108 whether the operator m
15 is a carrier. If the operator m is found a modulator, this operator computation
16 processing is immediately ended. If the operator m is found a carrier, the
17 waveform sample data OPOUTm generated in step S111 is multiplied by the
18 volume data VOLn of music tone parameter VOICE n. The result (= OPOUTm
19 x VOLn) is added to 2^f addresses of the buffer continued from the position
20 indicated by the pointer in the above-mentioned waveform buffer. Then, the
21 pointer is incremented by 2^f (step S114), upon which this operator computation
22 processing comes to an end. It should be noted that, when writing the plural
23 pieces of sample data of the same value in step S114, interpolation may be
24 made between the pieces of sample data as required, writing the resultant
25 interpolation value to the above-mentioned areas.

1 In the present embodiment, as explained in steps S106 and S111, the
2 values stored in the basic waveform table are used for the basic waveform data.
3 Alternatively, the basic waveform data may be generated by computation. Also,
4 the basic waveform data may be generated by combining table data and
5 computation. For the address by which the basic waveform table is read in
6 steps S106 and S110, the address obtained based on the phase value PHBUFm
7 computed in steps S105 and S110 is used. Alternatively, the address obtained
8 by distorting this phase value PHBUFm by computation or by a nonlinear
9 characteristic table may be used.

10 FIG. 20 is a flowchart indicating the detailed procedure of the timbre
11 setting processing subroutine of step S19 shown in FIG. 10. Referring to FIG.
12 20, first, MIDI channels and corresponding timbres are set (step S121). As
13 explained before, in the present embodiment, the MIDI channels and the
14 corresponding timbres are determined from the MIDI-CH voice table. The data
15 to be loaded into this MIDI-CH voice table is stored in the hard disk or the like.
16 When the MIDI file selected by the user is loaded, the corresponding table data
17 is loaded into the MIDI-CH voice table at the same time. Therefore, the
18 processing performed in step S121 is only the editing of the currently loaded
19 data table or the loading of new table data.

20 It should be noted that the user may alternatively set the desired number
21 of operators for each of MIDI channels. If the desired number of operators is
22 set to the channel concerned when changing the voice numbers in the MIDI-CH
23 voice table, the voice numbers corresponding to the music tone parameters
24 VOICE equal to or lower than the number of operators may be displayed in a
25 list. From among these voice numbers, the user may select and set desired ones.
26 At this time, the desired number of operators set to the channel concerned may

1 also be automatically changed. The voice numbers within the automatically
2 changed number of operators may be displayed in a list. Moreover, when the
3 user has changed the voice numbers in the MIDI-CH voice table, the total
4 number of operators constituting the music tone parameters VOICE
5 corresponding to the changed voice numbers may be checked. According to
6 the load state of the CPU 1, warning that this timbre cannot be assigned to the
7 channel concerned may be displayed. In addition to such a warning, the voice
8 number of the channel concerned may be automatically changed to the voice
9 number of an alternate timbre obtained by the smaller number of operators.

10 As described, the present embodiment is constituted such that the
11 number of operators for use in the FM computation processing can be flexibly
12 changed according to the capacity of the CPU 1, the operating environment of
13 the embodiment, the purpose of use, and the setting of processing.
14 Consequently, the novel constitution can adjust the load of the CPU 1 and the
15 quality of output music tone waveforms without restriction, thereby
16 significantly enhancing the degree of freedom of the sound source system in its
17 entirety. In the present embodiment FM tone generating is used for the music
18 tone waveform generation. It will be apparent that the present invention is also
19 applicable to a sound source that performs predetermined signal processing
20 such as AM (Amplitude Modulation) and PM (Phase Modulation) by
21 combining music tone waveform generating blocks. Further, the CPU load
22 mitigating method according to the invention is also applicable to a sound
23 source based on waveform memory reading and to a physical model sound
24 source in software approach. The present embodiment is an example of
25 personal computer application. It will be apparent that the present invention is
26 also easily applicable to amusement equipment such as game machines,

03920947 032997 466230 41602680

1 karaoke apparatuses, electronic musical instruments, and general-purpose
2 electronic equipment. Further, the present invention is applicable to a sound
3 source board and a sound source unit as personal computer options.

4 The software associated with the present invention may also be supplied
5 in disk media such as a floppy disk, a magneto-optical disk, and a CD-ROM, or
6 machine-readable media such as a memory card. Further, the software may be
7 added by means of a semiconductor memory chip (typically ROM) which is
8 inserted in a computer unit. Alternatively, the sound source software
9 associated with the present invention may be distributed through the
10 communication interface I/F 11. It may be appropriately determined according
11 to the system configuration or the OS whether the sound source software
12 associated with the present invention is to be handled as application software or
13 device software. The sound source software associated with the present
14 invention or the capabilities of this software may be incorporated in other
15 software; for example, amusement software such as game and karaoke and
16 automatic performance and accompaniment software.

17 The inventive machine readable media is used for a processor machine
18 including a CPU and contains program instructions executable by the CPU for
19 causing the processor machine having operators in the form of submodules
20 composed of softwares to compute waveforms for performing operation of
21 generating a plurality of musical tones through a plurality of channels
22 according to performance information. The operation comprises the steps of
23 setting an algorithm which determines a module composed of selective ones of
24 the submodules logically connected to each other to compute a waveform
25 specific to one of the musical tones, designating one of the channels to be used
26 for generating said one musical tone in response to the performance

1 information, loading the selective submodules into said one channel, and
2 logically executing the loaded selective submodules according to the algorithm
3 so as to compute the waveform to thereby generate said one musical tone
4 through said one channel.

5 Preferably, the step of setting sets different algorithms which determine
6 different modules corresponding to different timbres of the musical tones.
7 Each of the different modules is composed of selective ones of the submodules
8 which are selectively and sequentially connected to each other to compute a
9 waveform which is specific to a corresponding one of the different timbres.

10 Preferably, the step of setting comprises adjusting a number of the
11 submodules to be loaded into the channel dependently on a condition during
12 the course of generating the musical tone.

13 Preferably, the step of adjusting comprises compacting the module so as
14 to reduce the number of the submodules when the condition indicates that an
15 amplitude envelope of the waveform attenuates below a predetermined
16 threshold level.

17 Preferably, the step of adjusting comprises compacting the module so as
18 to reduce the number of the submodules when the condition indicates that an
19 output volume of the musical tone is tuned below a predetermined threshold
20 level.

21 Preferably, the step of adjusting comprises eliminating one submodule
22 so as to reduce the number of the submodules to be loaded into the channel
23 when the condition indicates that said one submodule loses contribution to
24 computation of the waveform without substantially affecting other
25 submodules.

The inventive machine readable media contains instructions for causing a processor machine having a software module to compute samples of a waveform in response to a sampling frequency for performing operation of generating a musical tone according to performance information. The operation comprises the steps of periodically operating the processor machine to execute the software module based on a variable sampling frequency for successively computing samples of the waveform so as to generate the musical tone, detecting a load of computation imposed on the processor machine during the course of generating the musical tone, and changing the variable sampling frequency according to the detected load to adjust a rate of computation of the samples.

Preferably, the step of changing provides a fast sampling frequency when the detected load is relatively light, and provides a slow sampling frequency when the detected load is relatively heavy such that the rate of the computation of the samples is reduced by $1/n$ where n denotes an integer number.

FIG. 21 shows a software sound source system practiced as a second preferred embodiment of the present invention. Referring to FIG. 21, a MIDI output section denoted by APS1 is a module for outputting a MIDI message. The APS1 is a performance operator device such as a keyboard, a sequencer for outputting a MIDI message, or application software for outputting a MIDI message. A MIDI API denoted by IF1 is a first application program interface that transfers MIDI messages to an operation system OS. A software sound source module SSM is application software installed in the operating system OS as a driver. The software sound source module SSM receives a MIDI message from the MIDI output section APS1 via the interface IF1. Based on

the received MIDI message, the software sound source module SSM generates tone waveform data. The generated tone waveform data is received by the operating system OS via a second application program interface (WAVE out API) IF2 of the OS. An output device OUD is a driver module installed in the operating system OS. The OUD receives the tone waveform data from the software sound source module SSM via the interface IF2, and outputs the received tone waveform data to external CODEC hardware. The output device OUD is software and operates in direct access memory (DMA) manner to read the tone waveform data which is generated by the computation by the software sound source module SSM and stored in a buffer. The OUD supplies the read tone waveform data to the external hardware composed of a digital-to-analog converter (DAC). The software sound source module SSM includes a tone generator for generating samples of tone waveform data at a predetermined sampling frequency FS by computation, and a MIDI output driver for driving this tone generator. This MIDI output driver reads tone control parameters corresponding to the received MIDI message from a table or the like, and supplies the read parameters to the tone generator.

FIG. 22 is a timing chart indicating the operation of the software sound source module SSM. As shown, the software sound source module SSM is periodically driven at every frame having a predetermined time length. In computation, the tone control parameters corresponding to the MIDI message received in an immediately preceding frame have been read and stored in a buffer. Based on the various tone parameters stored in the buffer, the SSM generates tone waveform. As shown in FIG. 22, the SSM receives three MIDI messages in a first frame from time T1 to time T2. When computation time T2 comes, the software sound source module SSM is started, upon which the

1 various parameters corresponding to the received MIDI messages are read and
2 stored in the buffer. Based on the received MIDI messages, the SSM performs
3 computation to generate tone waveform data to be newly sounded continuously
4 from the preceding frame.

5 In the computation for generating the tone waveform data, a number of
6 samples for one frame is generated for each sound channel. The tone waveform
7 data for all sound channels are accumulated and written to a waveform output
8 buffer. Then, reproduction of the waveform output buffer is reserved for the
9 output device OUD. This reservation is equivalent to outputting of the
10 generated tone waveform data from the software sound source module SSM to
11 the second interface "WAVE out API." The output device OUD reads, for each
12 frame, the tone waveform data a sample by sample from the waveform output
13 buffer reserved for reproduction, and sends the read tone waveform data to the
14 DAC which is the external hardware. For example, from the waveform output
15 buffer which is reserved for reproduction and written with the tone waveform
16 data generated in the first frame from time T1 to time T2, the tone waveform
17 data is read in the second frame from time T2 to Time T3. The read tone
18 waveform data is converted by the DAC into an analog music tone waveform
19 signal to be sounded from a sound system.

20 FIG. 23 outlines a processing apparatus having a tone waveform data
21 generator provided by implementing the tone waveform generating method
22 according to the invention. The processor shown in FIG. 23 uses a CPU 1 as
23 the main controller. Under the control of the CPU 1, the tone waveform
24 generating method according to the invention is executed as the tone waveform
25 generation processing based on a software sound source program. At the same
26 time, other application programs are executed in parallel. The CPU 1 is

1 or a wave table sound source. The digital signal processor board 9 is composed
2 of a DSP 9-1 for executing computation and a RAM 9-2 having various buffers
3 and various timbre parameters.

4 The network interface 11 connects this processing apparatus to the
5 Internet or the like via a LAN such as Ethernet or via a telephone line, thereby
6 allowing the processing apparatus to receive application software such as
7 sound source programs and data from the network. The MIDI interface 12
8 transfers MIDI messages between an external MIDI equipment and, receives
9 MIDI events from a performance operator device 13 such as a keyboard
10 instrument. The contents and reception times of the MIDI messages inputted
11 through this MIDI interface 12 are stored in the input buffer area of the RAM 2.

12 The CODEC 14 reads the tone waveform data from the waveform output
13 buffer of the RAM 3 in direct memory access manner, and stores the read tone
14 waveform data in a sample buffer 14-1. Further, the CODEC 14 reads samples
15 of the tone waveform data, one by one, from the sample buffer 14-1 at a
16 predetermined sampling frequency FS (for example, 44.1 kHz), and converts
17 the read samples through a DAC 14-2 into an analog music tone signal, thereby
18 providing a music tone signal output. This tone output is inputted into the
19 sound system for sounding. The above-mentioned constitution is generally the
20 same as that of a personal computer or a workstation. The tone waveform
21 generating method according to the present invention can be practiced by such
22 a machine.

23 The following outlines the tone waveform generating method according
24 to the present invention by means of the software sound source module under
25 the control of the CPU 1. When the application program APS1 is started, MIDI
26 messages are supplied to the software sound source module SSM via the first

1 interface IF1. Then, the MIDI output driver of the software sound source
2 module SSM is started to set tone control parameters corresponding to the
3 supplied MIDI messages. These tone control parameters are stored in sound
4 source registers of respective sound channels assigned with the MIDI messages.
5 Consequently, a predetermined number of samples of waveform data are
6 generated by computation in the sound source that is periodically activated
7 every computation frame as shown in FIG. 22.

8 FIGS. 24 through 26 show an example of a sound source model based on
9 the tone waveform data generating method according to the present invention.
10 It should be noted that this sound source model is implemented not by hardware
11 but by software. The sound source model illustrated in FIG. 24 through FIG.
12 26 simulates a wind instrument system or a string instrument system. This
13 model is hereafter referred to as a physical model sound source. The physical
14 model sound source of the wind instrument system simulates an acoustic wind
15 instrument having a mouthpiece at a joint of two tubes as shown in FIG. 24.
16 The physical model sound source of the string instrument system simulates a
17 plucked string instrument or a rubbed string instrument having strings fixed at
18 both ends with bridges.

19 The physical model sound source shown in FIG. 24 is composed of a
20 looping circuit. The total delay time in the loop corresponds to a pitch of a
21 music tone to be generated. When the physical model sound source simulates a
22 wind instrument, the sound source includes a circuit for simulating the tube
23 disposed rightward of the mouthpiece. In this circuit, a junction of 4-
24 multiplication grid type composed of four multipliers MU4 through MU7 and
25 two adders AD4 and AD5 simulates a tone hole. Further, a propagation delay
26 in the tube from the mouthpiece to the tone hole is simulated by a delay circuit

0802047 08297
466280 44602680

1 DELAY-RL. The propagation delay in the tube from the tone hole to the tube
2 end is simulated by a delay circuit DELAY-RR. Acoustic loss of the tube is
3 simulated by a lowpass filter FILTER-R. Reflection at the tube end is
4 simulated by a multiplier MU8. Similarly, in a circuit for simulating the tube
5 disposed leftward of the mouthpiece, the propagation delay of this tube is
6 simulated by a delay circuit DELAY-L. The acoustic loss of the tube is
7 simulated by a lowpass filter FILTER-L. The reflection at the tube end is
8 simulated by a multiplier MU3.

9 It should be noted that delay times DRL, DRR, and DL read from a table
10 according to the pitch of the music tone to be generated are set to the delay
11 circuits DELAY-R, DELAY-RR, and DELAY-L, respectively. Filter
12 parameters FRP and FRL for obtaining selected timbres are set to the lowpass
13 filters FILTER-R and FILTER-L, respectively. In order to simulate the
14 acoustic wave propagation mode that varies by opening or closing the tone hole,
15 multiplication coefficients M1 through M4 corresponding to the tone hole
16 open/close operations are supplied to the multipliers MU4 through MU7,
17 respectively. In this case, the pitch of the output tone signal is generally
18 determined by the sum of delay times to be set to the delay circuits DELAY-RL,
19 DELAY-RR, and DELAY L. Since an operational delay time occurs on the
20 lowpass filters FILTER-R and FILTER-L, a net delay time obtained by
21 subtracting this operation delay time is distributively set to the delay circuits
22 DELAY-RL, DELAY-RR, and DELAY-L in a .

23 The mouthpiece is simulated by a multiplier MU2 for multiplying a
24 reflection signal coming from the circuit for simulating the right-side tube by
25 multiplication coefficient J2 and a multiplier MU1 for multiplying a reflection
26 signal coming from the circuit for simulating the left-side tube by

08020947-082997
466280-466280

1 multiplication coefficient J1. The output signals of the multipliers MU1 and
2 MU2 are added together by an adder AD1, outputting the result to the circuits
3 for simulating the right-side tube and the circuit for simulating the left-side
4 tube. In this case, the reflection signals coming from the tube simulating
5 circuits are subtracted from the output signals by subtractors AD2 and AD3,
6 respectively, the results being supplied to the tube simulating circuits. An
7 exciting signal EX OUT supplied from an exciter and multiplied by coefficient
8 J3 is supplied to the adder D1. An exciter return signal EXT IN is returned to
9 the exciter via an adder AD6. It should be noted that the exciter constitutes a
10 part of the mouthpiece.

11 The output from this physical model sound source may be supplied to the
12 outside at any portion of the loop. In the illustrated example, the output signal
13 from the delay circuit DELAY-RR is outputted as an output signal OUT. The
14 outputted signal OUT is inputted into an envelope controller EL shown in FIG.
15 25, where the signal is attached with an envelope based on envelope parameters
16 EG PAR. These envelope parameters include a key-on attack rate parameter
17 and a key-off release rate parameter. Further, the output from the EL is inputted
18 into a resonator model section RE. The RE attaches resonance formant of the
19 instrument body to the signal based on the supplied resonator parameter. The
20 output signal from the EL is inputted into an effector EF. The EF attaches a
21 desired effect to a music signal TONE OUT based on supplied effect
22 parameters. The EF is provided for attaching various effects such as
23 reverberation, chorus, delay, and pan. The music tone signal TONE OUT is
24 provided in the form of samples of tone waveform data at every predetermined
25 sampling period.

FIG. 26 shows an example of the exciter that constitutes a part of the mouthpiece. The exciter return signal EX IN is supplied to a subtractor AD11 as a signal equivalent to the pressure of an air vibration wave to be fed back to the reed in the mouthpiece. From this signal, a blowing pressure signal P is subtracted. The output from the subtractor AD11 provides a signal equivalent to the pressure inside the mouthpiece. This signal is inputted into an exciter filter FIL10 simulating the response characteristics of the reed relating to pressure change inside the mouthpiece. At the same time, this signal is inputted into a nonlinear converter 2 (NLC2) simulating saturation characteristics of the velocity of the air flow inside the mouthpiece relating to the air pressure inside the mouthpiece when gain adjustment is performed by a multiplier MU11. A cutoff frequency of the exciter filter FIL10 is controlled selectively by a supplied filter parameter EF. The output signal from the exciter filter FIL10 is adjusted in gain by a multiplier MU10. The adjusted signal is added with an embouchure signal E equivalent to the mouthing pressure of the mouthpiece by an adder AD10, providing a signal equivalent to the pressure applied to the reed. The output signal from the adder AD10 is supplied to the nonlinear converter (NLC1) simulating the reed open/close characteristics. The output of the nonlinear converter 1 and the output of the nonlinear converter 2 are multiplied with each other by a multiplier MU12, from which a signal equivalent to the volume velocity of the air flow passing the gap between the mouthpiece and the reed is outputted. The signal outputted from the multiplier MU12 is adjusted in gain by a multiplier MU13, and is outputted as the exciting signal EX OUT.

The source model simulating a wind instrument has been explained above. In simulating a string instrument, a circuit for simulating a rubbed string section or a plucked string section in which a vibration is applied to a

1 string is used instead of the circuit for simulating the mouthpiece. Namely, the
2 signal P becomes an exciting signal corresponding to a string plucking force
3 and a bow velocity, and the signal E becomes a signal equivalent to a bow
4 pressure. It should be noted that, in simulating a string instrument, a
5 multiplication coefficient NL2G supplied to the multiplier MU11 is made
6 almost zero. Further, by setting the output of the nonlinear converter 2 to a
7 predetermined fixed value (for example, one), the capability of the nonlinear
8 converter 2 is not used. The delay circuits DELAY-RL, DELAY-RR, and
9 DELAY-L become to simulate string propagation times. The lowpass filters
10 FILTER-R and FILTER-L become to simulate string propagation losses. In the
11 exciter, setting of the multiplication coefficients NLG1, NLG2, NL1, and NL2
12 allows the exciter to be formed according to a model instrument to be
13 simulated.

14 The following explains various data expanded in the RAM 3 with
15 reference to FIG. 27. As described above, when the software sound source
16 module SSM is started, the MIDI output driver therein is activated, upon which
17 various tone control parameters are stored in the RAM according to the
18 inputted MIDI messages. Especially, if the MIDI messages designate a
19 physical model sound source (also referred to as a VA sound source) as shown
20 in FIGS. 24 through 26, a tone control parameter VATONEPAR for the
21 selected VA sound source is stored in the control parameter buffer
22 (VATONEBUF) arranged in the RAM 3. The tone waveform data generated
23 by computation by the software sound source module SSM for every frame is
24 stored in the waveform output buffer (WAVEBUF) in the RAM 3. Further, the
25 contents of each MIDI message inputted via the interface MIDI API and the
26 event time of reception of the inputted message are stored in MIDI input

1 buffers (MIDI RCV BUF and TM) in the RAM 3. Further, the RAM 3 has a
2 CPU work area.

3 The buffer VATONEBUF stores the tone control parameter
4 VATONEPAR as shown in FIG. 28. The VATONEBUF also stores a
5 parameter SAMPFREQ indicating an operation sampling frequency at which
6 samples of the tone waveform data are generated, a key-on flag VAKEYON
7 which is set when a key-on event contained in a MIDI message designates the
8 VA sound source, a parameter PITCH(VAKC) for designating a pitch, a
9 parameter VAVEL for designating a velocity when the key-on event designates
10 the VA sound source, and a breath controller operation amount parameter
11 BRETH CONT. Moreover, the VATONEBUF has a pressure buffer PBUF for
12 storing breath pressure and bow velocity, a PBBUF for storing a pitch bend
13 parameter, an embouchure buffer EMBBUF for storing an embouchure signal
14 or a bow pressure signal, a flag VAKONTRUNCATE for designating sounding
15 truncate in the VA sound source, and a buffer miscbuf for storing volume and
16 other parameters.

The parameter SAMPFREQ can be set to one of two sampling frequencies, for example. The first sampling frequency is 44.1 kHz and the second sampling frequency is a half of the first sampling frequency, namely 22.05 kHz. Alternatively, the second sampling frequency may be double the first sampling frequency, namely 88.2 kHz. These sampling frequencies are illustrative only, hence not limiting the sampling frequencies available in the present invention. Meanwhile, if the sampling frequency is reduced 1/2 times FS, the number of the tone waveform samples generated in one frame may be reduced by half. Consequently, if the load of the CPU 1 is found heavy, the

59

1 sampling frequency of 1/2 times FS may be selected to mitigate the load of the
2 CPU 1, thereby preventing dropping of samples from generation.

3 If the sampling frequency is set to 2 times FS, the number tone waveform
4 samples generated is doubled, allowing the generation of high-precision tone
5 waveform data. Consequently, if the load of the CPU 1 is found light, the
6 sampling frequency of 2 times FS may be selected to generate samples having
7 high-precision tone waveform data. For example, let the standard sampling
8 frequency in the present embodiment be FS1, a variation sampling frequency
9 FS2 is represented by:

10 $FS1 = n \text{ times } FS2$ (n being an integer) ... first example,

11 $FS1 = 1/n \text{ times } FS2$ (n being an integer) ... second example.

12 Because the present invention mainly uses the first example, the following
13 description will be made mainly with reference to the first example.

14 In the present invention, the sampling frequencies of the tone waveform
15 data to be generated are variable. If there is another acoustic signal to be
16 reproduced by the CODEC, the sampling frequency of the DA converter in the
17 CODEC may be fixed to a particular standard value. For example, when
18 mixing the music tone generated by the software sound source according to the
19 present invention with the digital music tone outputted from a music CD, the
20 sampling frequency may be fixed to $FS1 = 44.1 \text{ kHz}$ according to the standard
21 of the CD. The following explains an example in which the sampling
22 frequency of the CODEC is fixed to a standard value. The relation between this
23 standard sampling frequency FS1 and the variation sampling frequency FS2 is
24 represented by $FS1 = n \text{ times } FS2$ as described before. The sampling frequency
25 of the DA converter is fixed to the standard value. Therefore, it is required for
26 the waveform output buffer WAVEBUF which is read a sample by sample

1 every period of this fixed standard sampling frequency FS1 to store beforehand
2 a series of the waveform data in matching with the standard sampling frequency
3 FS1 regardless of the sampling frequency selected for the waveform
4 computation. If the sampling frequency FS2 which is $1/n$ of the sampling
5 frequency FS1 is selected, the resultant computed waveform samples are
6 written to the waveform output buffer WAVEBUF such that n samples of the
7 same value are arranged on continuous buffer addresses. When the waveform
8 data for one frame has been written to the waveform output buffer WAVEBUF,
9 the contents of the waveform buffer WAVEBUF may be passed to the CODEC.
10 Since the sampling frequency FSb of the data series stored in the waveform
11 output buffer WAVEBUF differs from the operation sampling frequency FSc
12 of the CODEC (or DAC), sampling frequency matching may be required. For
13 example, if $FSb = k$ times FSc ($K > 1$), then the tone waveform data may be
14 sequentially passed from the waveform output buffer WAVEBUF in skipped
15 read manner by updating every n addresses. Namely, during the time from the
16 processing of storing the music waveform samples in the waveform output
17 buffer WAVEBUF to the processing of the DAC of the CODEC, a sampling
18 frequency conversion circuit may be inserted to match the write and read
19 sampling frequencies.

20 Information about the time at which storage is made in the MIDI event
21 time buffer TM is required for performing the time-sequential processing
22 corresponding to occurrence of note events. If the frame time is set to a
23 sufficiently short value such as 5 ms or 2.5 ms, adjustive fine timing control for
24 various event processing operations is not required substantially in the frame ,
25 so that these event processing operations need not be performed by especially
26 considering the time information. However, it is preferable that the

1 information from the breath controller and so on be handled on a last-in first-
2 out basis, so that, for the event of this information, processing on the last-in
3 first-out basis is performed by use of the time information. In addition to the
4 above-mentioned buffers, the RAM 3 may store application programs.

5 FIG. 28 shows details of the tone control parameters VATONEPAR.
6 The tone control parameters VATONPAR include an exciter parameter
7 (EXCITER PARAMETERS), a wind instrument/string instrument parameter
8 (P/S PARAMETERS), an envelope parameter (EG PAR), a resonator
9 parameter (RESONATOR PAR), an effect parameter (EFFECT PAR), and
10 sampling frequency data (SAMPLING FREQ). Each of these parameters
11 includes a plurality of parameter items. Each delay amount parameter and each
12 tone hole junction multiplication coefficient are determined by a pitch of a
13 musical tone. In this case, DL through DRR are tables listing a delay amount
14 for a pitch. Delay amounts are read from these tables and set so that a total
15 delay amount corresponds to a desired pitch. Each of these delay amount tables
16 is prepared by actually sounding a tone having a predetermined pitch and by
17 feeding back a deviation in the pitch frequency. The filter parameters such as
18 FLP and FRP are set according to the contour of the tube to be simulated, the
19 characteristics of the string, and the operation amount of the operator device. It
20 should be noted that preferred tone control parameters VATONEPAR are set
21 according to the sampling frequency used. The sampling frequency of these
22 tone control parameters VATONEPAR is indicated by SAMPLING FREQ in
23 FIG. 28. The processing for waveform generation by computation is performed
24 by using the tone control parameters VATONEPAR prepared for the sampling
25 frequency concerned by referencing this SAMPLING FREQ information. In

1 this example, the standard sampling frequency is FS1 and the alternative
2 sampling frequency FS2 is 1/2 times FS1, for example.

3 As described above, the inventive sound source apparatus has a software
4 module used to compute samples of a waveform in response to a sampling
5 frequency for generating a musical tone according to performance information.
6 In the inventive apparatus, a processor device composed of the CPU 1
7 periodically executes the software module SSM for successively computing
8 samples of the waveform corresponding to a variable sampling frequency so as
9 to generate the musical tone. A detector device included in the CPU 1 detects a
10 load of computation imposed on the processor device during the course of
11 generating the musical tone. A controller device implemented by the CPU 1
12 operates according to the detected load for changing the variable sampling
13 frequency to adjust a rate of computation of the samples.

14 Preferably, the controller device provides a fast sampling frequency
15 when the detected load is relatively light, and provides a slow sampling
16 frequency when the detected load is relatively heavy such that the rate of the
17 computation of the samples is reduced by $1/n$ where n denotes an integer
18 number.

The processor device includes a delay device having a memory for imparting a delay to the waveform to determine a pitch of the musical tone according to the performance information. The delay device generates a write pointer for successively writing the samples into addresses of the memory and a read pointer for successively reading the samples from addresses of the memory to thereby create the delay corresponding to an address gap between the write pointer and the read pointer..The delay device is responsive to the fast sampling frequency to increment both of the write pointer and the read pointer by one

63

1 address for one sample. Otherwise, the delay device is responsive to the slow
2 sampling frequency to increment the write pointer by one address n times for
3 one sample and to increment the read pointer by n addresses for one sample.

4 The processor device may include a delay device having a pair of
5 memory regions for imparting a delay to the waveform to determine a pitch of
6 the musical tone according to the performance information. The delay device
7 successively writes the samples of the waveform of one musical tone into
8 addresses of one of the memory regions, and successively reads the samples
9 from addresses of the same memory region to thereby create the delay. The
10 delay device is operative when said one musical tone is switched to another
11 musical tone for successively writing the samples of the waveform of said
12 another musical tone into addresses of the other memory region and
13 successively reading the samples from addresses of the same memory region to
14 thereby create the delay while clearing the one memory region to prepare for a
15 further musical tone.

16 Preferably, the processor device executes the software module composed
17 of a plurality sub-modules for successively computing the waveform. The
18 processor device is operative when one of the sub-modules declines to become
19 inactive without substantially affecting other sub-modules during computation
20 of the waveform for skipping execution of said one sub-module.

21 The inventive sound source apparatus has a software module used to
22 compute samples of a waveform for generating a musical tone. In the inventive
23 apparatus, a provider device variably provides a trigger signal at a relatively
24 slow rate to define a frame period between successive trigger signals, and
25 periodically provides a sampling signal at a relatively fast rate such that a
26 plurality of sampling signals occur within one frame period. The processor

device is resettable in response to each trigger signal and is operable based on each sampling signal to periodically execute the software module for successively computing a number of samples of the waveform within one frame. The detector device detects a load of computation imposed on the processor device during the course of generating the musical tone. The controller device is operative according to the detected load for varying the frame period to adjust the number of the samples computed within one frame period. A converter device composed of CODEC 14 is responsive to each sampling signal for converting each of the samples into a corresponding analog signal to thereby generate the musical tones.

The following explains the operations of the present invention in detail with reference to flowcharts.

FIG. 29 is a flowchart showing an initialization program to be executed at a power-on or reset sequence. When the initialization program is started, system initialization such as hardware initialization is performed in step SS10. Next, in step SS11, the OS program is started to place other programs in an executable state in which a main program for example is executed.

FIG. 30 is a flowchart showing the main program to be executed by the CPU 1. When the main program is started, initialization such as resetting of registers is performed in step SS20. Next, in step SS21, basic display processing such as arranging windows for display screens such as desktop is performed. Then, in step SS22, trigger check is performed for task switching. In step SS23, it is determined whether a trigger has taken place. The operations of steps SS21 through SS23 are repeated cyclically until a trigger is detected. If a trigger is found, the decision in step SS23 turns YES. In step SS24, the task

65

08920947 082997
455230 44602650

1 switching is performed so that a task corresponding to the detected trigger is
2 executed.

3 There are five types of triggers for commencing the task switching. If
4 supply of a MIDI message from an application program or the like via the
5 sound source API (MIDI API) is detected, it indicates trigger 1. In this case, the
6 software sound source module SSM is started in step SS25 to perform MIDI
7 processing. If an internal interrupt has been caused by a software timer (tim)
8 that outputs the interrupt every frame period, it indicates trigger 2. In this case,
9 the software sound source module SSM is started in step SS26 to perform
10 waveform computation processing, thereby generating tone waveform data for
11 the predetermined number of samples. If a transfer request for tone waveform
12 data has been made by an output device (CODEC) based on DAM, it indicates
13 trigger 3. In this case, transfer processing is performed in step SS27 in which
14 the tone waveform data is transferred from the waveform output buffer
15 WAVEBUF to the output device. If an operation event based on manual
16 operation of the input operator device such as the mouse or the keyboard of the
17 processing apparatus has been detected, it indicates trigger 4. In the case of the
18 operation event for timbre setting, timbre setting processing is performed in
19 step SS28. For other operation events, corresponding processings are
20 performed in step SS29. If the end of the operation has been detected, it
21 indicates trigger 5. In this case, end processing is performed in step S30. If no
22 trigger has been detected, trigger 4 is assumed and the processing of steps SS28
23 and SS29 is performed. When the processing of trigger 1 to trigger 5 has been
24 completed, the SSM returns control to step SS21. The processing operations of
25 steps SS21 through SS30 are repeated cyclically.

FIG. 31 is a flowchart showing the MIDI processing to be performed in step SS25. When the MIDI processing is started, the contents of the MIDI event are checked in step S40. This check is specifically performed on the MIDI message written to "MIDI API" constituted as a buffer. Then, it is determined in step SS41 whether the MIDI event is a note-on event. If the MIDI event is found a note-on event, the SSM passes control to step SS42, in which it is determined whether the sound channel (MIDI CH) assigned to that note-on event belongs to a physical model sound source or a VA sound source. If the sound channel assigned to the note-on event is found in the physical model sound source (hereafter, such a MIDI CH is labeled "VA CH"), the key-on processing in the physical model sound source is performed in step SS43 and control is returned. If the sound channel assigned to the note-on event is not found in the physical model sound source, the key-on processing of another sound source is performed in step SS44, upon which control is returned. This key-on processing is performed in the DSP 9-1 of the digital signal processing board 9, for example.

If the MIDI event is found not a note-on event in step SS41, it is determined in step SS45 whether the MIDI event is a note-off event. If the MIDI event is found a note-off event, it is determined in step SS46 whether the sound channel (MIDI CH) assigned to the note-off event belongs to the physical model sound source. If the sound channel assigned to the note-off event is found in the physical model sound source, the key-on flag VAKEYON in the physical model sound source is set to "0" in step SS47, and the occurrence time of the note-off event is stored in the MIDI event time buffer TM, upon which control is returned. If the sound channel assigned to the note-off event is not found in the physical model sound source, the key-off

1 processing of another sound source is performed in step SS48, upon which
2 control is returned.

3 Further, if the MIDI event is found not a key-off event in step SS45, it is
4 determined in step SS 49 whether the MIDI event is a program change. If the
5 MIDI event is found the program change, it is determined in step SS50 whether
6 the sound channel (MIDI CH) assigned to the MIDI event of program change
7 belongs to the physical model sound source. If the sound channel assigned to
8 the MIDI event of program change is found in the physical model sound source,
9 the tone control parameters VATONEPAR designated in the program change
10 are stored in step SS51, upon which control is returned. If the sound channel
11 assigned to the MIDI event of program change is not found in the physical
12 model sound source, the timbre parameter processing corresponding to that
13 sound channel is performed in step SS52, upon which control is returned. If the
14 MIDI event is not a program change in step SS49, the processing of the
15 corresponding MIDI event is performed in step SS53, upon which control is
16 returned. In this MIDI event processing, the processing for a breath controller
17 operation is performed, for example.

18 FIG. 32A is a flowchart showing the key-on processing of the physical
19 model sound source to be performed in step SS43. When the physical model
20 sound source key-on processing is started, the note number contained in the
21 received MIDI message is stored in the buffer VATONEBUF as a parameter
22 VAKC in step SS55. The velocity information contained in the same MIDI
23 message is stored in the VATONEBUF as a parameter VAVEL. The
24 VAKEYON flag is set to "1". Further, the MIDI message receive time is stored
25 in the buffer TM as an event occurrence time. Pitch frequency data converted
26 from the parameter VAKC and the pitch bend value stored in the pitch bend

1 buffer PBBUF are stored in the buffer VATONEBUF as a parameter PITCH.
2 When these processing operations come to an end, control is returned. It
3 should be noted that, instead of using the pitch bend value for obtaining the
4 pitch frequency, the pitch bend value may be used for setting an embouchure
5 parameter.

6 FIG. 32B is a flowchart showing the timbre setting processing to be
7 performed in step SS28 when the above-mentioned trigger 4 has been detected.
8 When the user performs a timbre setting operation by manipulating the mouse
9 or keyboard, the timbre setting processing is started. In step SS50, it is
10 determined whether timbre setting of the physical model sound source has been
11 designated. If the timbre setting is found designated, the timbre parameter
12 corresponding to the designated timbre is expanded in the buffer
13 VATONEBUF as shown in FIG. 27 in step SS61. Then, in step SS62, the
14 timbre parameter is edited by the user, upon which the timbre setting
15 processing comes to an end. If the timbre setting is found not designated in step
16 SS60, control is passed to step SS62, in which the timbre parameter is edited by
17 the user and the timbre setting processing comes to an end.

18 FIG. 32C is a flowchart showing other MIDI event processing to be
19 performed in step SS53. When the other MIDI event processing is started, it is
20 determined in step SS65 whether the sound channel (MIDI CH) assigned to the
21 MIDI event belongs to the physical model sound source. If the sound channel
22 assigned to the MIDI event is found in the physical model sound source, it is
23 determined in step SS66 whether the MIDI event is a breath control event. If
24 the MIDI event is found a breath control event, the parameter BRETH CONT in
25 the breath control event is stored in the pressure buffer PBUF in step SS67.

If the MIDI event is found not a breath control event, step SS67 is skipped, and, in step SS68, it is determined whether the MIDI event is a pitch bend event. If the MIDI event is found a pitch bend event, it is determined in step SS69 whether the embouchure mode is set. If the embouchure mode is set, the parameter PITCHBEND in the pitch bend event is stored in the embouchure buffer EMBBUF in step SS70. If the embouchure mode is not set, the parameter PITCHBEND in the pitch bend event is stored in the pitch bend buffer PBBUF in step SS72.

Further, if it is found that the sound channel does not belong to the physical model sound source in step SS65 and if the MIDI event is found not a pitch bend event in step SS68, control is passed to step SS71, in which it is assumed that the received MIDI event does not correspond to any of the above-mentioned events, then processing corresponding to the received event is performed, and control is returned. It should be noted that the embouchure signal indicates a pressure with which the player mouths the mouthpiece. Since the pitch varies based on this embouchure signal, the parameter PITCHBEND is stored in the embouchure buffer EMBBUF in the embouchure mode. As described, every time a MIDI event is received, the parameters associated with music performance are updated by the MIDI event processing.

FIG. 33 is a flowchart showing the physical model parameter expanding processing. This processing is performed in step SS61 of the above-mentioned timbre setting processing before sounding. When the physical model parameter expanding processing is started, the CPU load state is checked in step SS75. This check is performed based on a status report from the CPU 1 for example and by considering the setting value of the sampling frequency FS. If this check indicates in step SS76 that the load of the CPU 1 is not yet heavy, the

1 shortest frame period of one frame set by the user or the standard frame period
2 TIMDEF is set in step SS77 as a period tim of the software timer that causes a
3 timer interrupt for conducting the waveform generation processing every frame.
4 It should be noted that the standard frame period TIMDEF is set to 2.5ms, for
5 example.

6 In step SS78, the sampling frequency FS specified by the tone control
7 parameter VATONEPAR for the selected physical model sound source is set as
8 the operation sampling frequency SAMPFREQ. Further, in step SS79, alarm
9 clear processing is performed. In step SS80, the tone control parameters
10 VATONEPAR containing to the parameter SAMPFREQ and the parameter
11 VAKC are read to be stored in the buffer VAPARBUF, upon which control is
12 returned. In this case, the tone control parameters VATONEPAR considering
13 the parameter VAVEL may be stored in the buffer VAPARBUF.

14 If the load of the CPU 1 is found heavy in step SS76, it is determined in
15 step SS81 whether the frame time automatic change mode is set. If this mode is
16 set, a value obtained by multiplying the standard frame period TIMDEF by
17 integer α is set as the period tim of the software timer in step SS82. Integer α is
18 set to a value higher than one. When the frame period is extended, the
19 frequency at which parameters are loaded into the physical model sound source
20 can be lowered, thereby reducing the number of processing operations for
21 transferring the changed data and the number of computational operations
22 involved in the data updating.

23 In step SS83, the current operation sampling frequency SAMPFREQ is
24 checked. If the operation sampling frequency SAMPFREQ is the sampling
25 frequency FS1, it indicates that the load of the CPU 1 is heavy, so that the
26 sampling frequency FS2 which is 1/2 of FS1 is set as the operation sampling

11

1 the tone control parameters VATONEPAR stored in the buffer VAPARBUF,
2 physical model computation processing is performed. Every time this
3 computation processing is performed, the tone waveform data for one sample is
4 generated. The generated tone waveform data is stored in the waveform output
5 buffer WAVEBUF.

6 In step SS111, it is determined whether the waveform computation for
7 the number of samples calculated in step SS91 has ended. If the computation is
8 found not ended, control is passed to step SS113, in which the time occupied by
9 computation by the CPU 1 in one frame time or a predetermined time is
10 checked. If this check indicates that the occupation time does not exceed the
11 one frame time, next sample computation processing is performed in step
12 SS110. The processing operations of steps SS110, SS111, SS113, and SS114
13 are cyclically performed until the predetermined number of samples is obtained
14 as long as the occupation time does not exceed the one frame time.
15 Consequently, it is determined in step SS111 that the computation of the
16 predetermined number of samples in one frame has ended. Then, in step SS112,
17 the tone waveform data stored in the waveform output buffer WAVEBUF is
18 passed to the output device (the CODEC).

19 If it is determined in step SS114 that one frame time has lapsed before
20 the predetermined number of samples has been computed, then, in step SS115,
21 the muting processing of the tone waveform data in the waveform output buffer
22 WAVEBUF is performed. Next, in step SS112, the tone waveform data stored
23 in the waveform output buffer WAVEBUF is passed to the output device (the
24 CODEC). If, in step SS90, the key-on flag VAKEYON is found not to set "1",
25 it is determined in step SS103 whether key-off processing is on. If the decision

1 is YES, the key-off processing is performed in step SS104. If the key-off
2 processing is found not on, control is returned immediately.

3 According to the invention, the tone generating method uses a hardware
4 processor in the form of the CPU 1 and a software module in the form of the
5 sound source module SSM to compute samples of a waveform in response to a
6 sampling frequency for generating a musical tone according to performance
7 information. The inventive method comprises the steps of periodically
8 operating the hardware processor to execute the software module for
9 successively computing samples of the waveform corresponding to a variable
10 sampling frequency so as to generate the musical tone, detecting a load of
11 computation imposed on the hardware processor during the course of
12 generating the musical tone, and changing the variable sampling frequency
13 according to the detected load to adjust a rate of computation of the samples.
14 Preferably, the step of changing provides a fast sampling frequency when the
15 detected load is relatively light, and provides a slow sampling frequency when
16 the detected load is relatively heavy such that the rate of the computation of the
17 samples is reduced by $1/n$ where n denotes an integer number.

18 The inventive method uses a hardware processor having a software
19 module used to compute samples of a waveform for generating a musical tone.
20 The inventive method comprises the steps of variably providing a trigger signal
21 at a relatively slow rate to define a frame period between successive trigger
22 signals, periodically providing a sampling signal at a relatively fast rate such
23 that a plurality of sampling signals occur within one frame period, operating the
24 hardware processor resettable in response to each trigger signal and operable in
25 response to each sampling signal to periodically execute the software module
26 for successively computing a number of samples of the waveform within one

1 frame, detecting a load of computation imposed on the software processor
2 during the course of generating the musical tone, varying the frame period
3 according to the detected load to adjust the number of the samples computed
4 within one frame period, and converting each of the samples into a
5 corresponding analog signal in response to each sampling signal to thereby
6 generate the musical tones.

7

8 Meanwhile, in order to build the physical model sound source in which
9 the sampling frequency is variable, a delay device is required in which the
10 sampling frequency is variable while a delay time can be set without restriction
11 from the sampling frequency. The following explains such a delay device with
12 reference to FIG. 38. In the physical model sound source, each delay circuit
13 uses a delay area in the RAM 3 as a shift register to obtain a predetermined
14 delay amount. A DELAYx 20 shown in FIG. 38 is the delay circuit constituted
15 by the delay area allocated in the RAM 3. The integer part of the delay amount
16 provides the number of shift register stages D between a write pointer
17 indicating an address location at which inputted data is written and a read
18 pointer indicating an address location at which the data is read. The decimal
19 fraction of the delay amount provides multiplication coefficient d to be set to a
20 multiplier MU21 to perform interpolation between a pair of the data read at an
21 address location indicated by the read pointer and the data read at an address
22 location (READ POINTER-n) n stages before that read pointer. It should be
23 noted that a multiplication coefficient (1 - d) is set to a multiplier MU20 for
24 interpolation.

25 In this case, a total delay amount of the delay outputs of an adder AD20
26 in the DELAYx 20 becomes (D + d) equivalent to the number of delay stages.

1 In the equivalent of time, the total delay amount becomes $(D + d)/FS$ for the
2 sampling frequency FS . If the maximum value among the sampling
3 frequencies is $FS1$, then it is desired to constitute the delay such that the
4 periodic time of the sampling frequency $FS1$ basically corresponds to one stage
5 of the delay circuit. In such a constitution, in order to lower the sampling
6 frequency to $1/n$ of the $FS1$, one sample obtained by the computation may be
7 written to n continuous stages of the delay circuit at n continuous addresses for
8 each sample computation. On the other hand, the delay outputs may be read by
9 updating the read pointer by n addresses. Therefore, in the above-mentioned
10 constitution, the equivalent value of the number of delay stages $(D + d)$ for
11 implementing necessary delay time T_d is $(D + d) = T_d \text{ times } FS1$ regardless of
12 the sampling frequency. It should be noted that the write pointer and the read
13 pointer are adapted to equivalently shift in the address direction indicated by
14 arrow on the shift register. When the pointers reach the right end of the shift
15 register, the pointers jump to the left end, thus circulating on the $DELAY \times 20$.

As described, since the delay time length of the time equivalent of one stage of delay is made constant ($1/FS1$) regardless of the sampling frequency FS , the write pointer is set to write one sample of the waveform data over continuous n addresses to maintain the delay time length of the delay output even if the sampling frequency FS is changed to the sampling frequency $FS2$ which is $1/n$ of $FS1$. Every time one sample of the waveform data is generated, the write pointer is incremented by n addresses. The read pointer is updated in units of n addresses ($n - 1$) at once to read the sample delayed by address skipping. This constitution allows the delay output the one sample of the generated waveform data to correspond to the delay output read from the address location before n addresses. Therefore, for the decimal fraction delay

1 part shown in FIG. 38, data before one sample for interpolation is read from an
2 address location n stages (n addresses) before the read pointer.

3 Also, in a unit delay means provided for a filter and so on in the physical
4 model sound source, a means generally similar to the above-mentioned delay
5 circuit is used to prevent the delay time length from being changed even if the
6 preset sampling frequency is changed. The following explains this unit delay
7 means with reference to FIG. 39. The unit delay means also uses the delay area
8 in the RAM 3 as a shift register. A DELAYx 21 shown in FIG. 39 is the unit
9 delay means composed of the delay area allocated in the RAM 3. The unit
10 delay amount of this means is obtained by the shift register through n stages
11 between an address location indicated by a write pointer to which data is
12 written and an address location indicated by a read pointer from which data is
13 read.

14 As described with the delay circuit shown in FIG. 38, one sample is
15 written into n consecutive addresses (n stages). Therefore, the address
16 difference between the write pointer and the read pointer is n addresses. In this
17 case, the write pointer is set such that the same value of one sample is written
18 over n addresses. The read pointer is set such that data is read by updating the
19 read pointer in units of n addresses. It should be noted that the unit delay means,
20 by nature, may be constituted only by n stages of delay areas.

21 The inventive sound source apparatus has a software module used to
22 compute samples of a waveform in response to a sampling frequency for
23 generating a musical tone according to performance information. In the
24 inventive apparatus, a processor device responds to a variable sampling
25 frequency to periodically execute the software module for successively
26 computing samples of the waveform so as to generate the musical tone. A

1 VATONEPAR_m(FS1, FS2), the parameters having the same timbre for each of
2 the sampling frequencies FS1 and FS2 are prepared all in one tone control
3 parameter VATONEPAR_i. In this case, the timbre parameter corresponding to
4 the sampling frequency FS is extracted from one tone control parameter
5 VATONEPAR_i, and the extracted parameter is stored in the buffer
6 VAPARBUF. The tone control parameters having the voice numbers
7 subsequent to VATONEPAR_m + 1 are the tone control parameters having
8 independent timbres corresponding to one of the sampling frequency FS1 and
9 the sampling frequency FS2. Namely, VATONEPAR_{m+1}(FS1, *) corresponds
10 only to the sampling frequency FS1. VATONEPAR_p(*, FS2) corresponds
11 only to the sampling frequency FS2. In order to prevent changing of the
12 sampling frequency from affecting uniqueness of the tone in terms of auditory
13 sensation, the parameters to be adjusted according to the changed sampling
14 frequency include the delay parameters of the delay loop section, the filter
15 coefficients, and the nonlinear characteristics of the nonlinear converter of the
16 exciter.

17 FIG. 36 is a flowchart of the physical model sound source processing to
18 be performed in step SS110 of the above-mentioned waveform generation
19 processing. When the physical model sound source processing is started, the
20 delay length setting processing of each variable delay section is performed in
21 step SS120 according to the designated pitch frequency, the operation sampling
22 frequency SAMPFREQ indicating the setting state of each section, and the tone
23 control parameter VATONEPAR stored in the buffer VAPARBUF. Each delay
24 time length is set as shown in FIG. 38. Then, in step SS121, the computation
25 processing associated with the exciter as shown in FIG. 26 is performed based
26 on the operation sampling frequency SAMPFREQ, the signal P of breath

0822947-0822947

1 pressure or bow velocity, the signal E of embouchure or bow pressure, and the
2 tone control parameter VATONEPAR stored in the buffer VAPARBUF.
3 Namely, the exciter return signal EX IN is captured. Then, based on the filter
4 parameter FLTPAR corresponding to the operation sampling frequency
5 SAMPFREQ, filter computation of the exciter filter FIL10 is performed.
6 Further, computation of the nonlinear converter 1 is performed by the nonlinear
7 conversion characteristics corresponding to the operation sampling frequency
8 SAMPFREQ. If required, computation of the nonlinear converter 2 is
9 performed. Also, computation of portions peripheral to these converters is
10 performed. Then, the exciter output signal EX OUT is generated and outputted.

11 In step SS122, computation processing associated with the tube/string
12 model shown in FIG. 24 is performed based on the operation sampling
13 frequency SAMPFREQ and the parameter VATONEPAR stored in the buffer
14 VAPARBUF. Namely, the exciter output signal EX OUT is captured, and
15 computation of the junction section is performed based on the junction
16 parameter JUNCTPAR corresponding to the operation sampling frequency
17 SAMPFREQ. Further, computation of the delay loop section is performed.
18 Based on the filter parameter FLTPAR corresponding to the operation
19 sampling frequency SAMPFREQ, computations of the terminal filters
20 FILTER-R and FILTER-L are also performed. Then, the generated exciter
21 return signal EX IN and the output sample signal OUT are outputted.

22 In step SS123, computation of the timbre effector as shown in FIG. 25 is
23 performed based on the operation sampling frequency SAMPFREQ and the
24 parameter VATONEPAR stored in the buffer VAPARBUF. Namely, the
25 output sample signal OUT is taken out, and computations of the envelope
26 controller EL, the resonator model section RE, and the effector EF are

83

1 performed, respectively. Then, the generated final output is outputted as the
2 tone waveform data TONEOUT. This tone waveform data TONEOUT is
3 written into the waveform output buffer WAVEBUF in response to the
4 sampling frequency FS as described above.

5 FIG. 37 is a flowchart of the delay loop computation processing
6 performed in step SS122 of the physical model section computation processing.
7 This flowchart shows in detail only the computation processing associated with
8 the terminal filter FILTER-R and the multiplier MU8. The computation
9 processing of the FILTER-L and the multiplier MU3 is performed in the same
10 manner. When the delay loop computation processing is started, computation
11 of the loop up to the right-side end immediately before the terminal filter
12 FILTER-R is performed in step SS130. Then, the computation skip condition
13 is checked in step SS131. This check is performed to skip the computation of
14 the section of which loop gain is substantially zero, thereby saving the total
15 computation amount. Specifically, there are three computation skip conditions.
16 The first computation skip condition is that the output of the terminal filter
17 FILTER-R is 0. This condition may also be that the value 0 is continuously
18 outputted from the terminal filter FILTER-R for a predetermined time. Further,
19 the input of the terminal filter FILTER-R and the contents of the internal delay
20 register may be checked. This condition may also be satisfied when the final
21 output TONEOUT is sufficiently attenuated. The second computation skip
22 condition is that the input signal of the terminal filter FILTER-R is not
23 substantially changed. In this case, the computation is skipped and the output
24 value from the immediately preceding terminal filter FILTER-R is assumed to
25 be the current output value. Further, the immediately preceding output value
26 may be the current output value also in the multiplier MU8. The third

1 computation skip condition is that the multiplication coefficient TERMGR of
2 the multiplier MU8 is zero or nearly zero. In this case, the computation is
3 skipped and the right-side output is made zero.

4 When any of the above-mentioned computation skip conditions that is
5 associated with the terminal filter FILTER-R has been satisfied, the decision is
6 made YES in step SS132. Then, in step SS133, processing for passing the
7 output value corresponding to the satisfied condition is performed. If the
8 computation skip condition associated with the terminal filter FILTER-R is
9 found not satisfied, the computation associated with the terminal filter
10 FILTER-R is performed in step SS137. When the processing in step SS133 or
11 SS137 has been completed, it is determined in step SS134 whether the
12 computation skip condition associated with the multiplication coefficient
13 TERMGR is satisfied. If this condition is found satisfied, the decision is YES.
14 Then, in step SS135, the processing for passing the output value corresponding
15 to the satisfied condition is performed. If the condition is found not satisfied,
16 computation for multiplying the multiplication coefficient TERMGR in the
17 multiplier MU8 is performed in step SS138. When the processing of step
18 SS135 or SS138 has been completed, computation processing of the remaining
19 delay loop portions is performed in step SS136, upon which control is returned.

20 Computation may be skipped not only with the delay loop but also with
21 the exciter or the timbre effector. For the exciter, whether the computation is to
22 be skipped or not is determined by checking the signal amplitude of the signal
23 path and the associated parameters if the values of the amplitude and the
24 parameters are nearly zero. For the timbre effector, when the output of the
25 envelope controller EL, the resonator model section RE, or the effector EF has
26 been sufficiently attenuated to nearly zero, the computation for each block of

1 which output is nearly zero may be skipped to make the output value zero. In
2 the second embodiment described so far, control of changing the sampling
3 frequency FS may cause an aliening noise depending on the nonlinear
4 conversion characteristics in the nonlinear section. This problem may be
5 overcome by performing over-sampling on the input side of the nonlinear
6 conversion and by band-limiting the obtained nonlinear conversion output by a
7 filter to return the sampling frequency to the original sampling frequency.

8 If a new key-on occurs during the current key-on state in the physical
9 model sound source shown in FIG. 24, processing for sounding the music tone
10 corresponding to the new key-on is performed. If the sounding is made by
11 inheriting the music tone corresponding to the preceding key-on, the signals
12 that circulate inside the physical model, for example, the signals inside the
13 delay sections such as the tube/string model section may be basically handled
14 without change. New exciter signals may only be generated according to the
15 new key-on. If a highly independent music tone is set up without making such
16 inheritance, or a music tone having a timbre different from that of the
17 immediately preceding key-on is to be sounded in response to the new key-on,
18 the delay circuit in the physical model sound source must be initialized or reset
19 according to the new key-on. In this case, if the number of sound channels in
20 the physical model sound source is one, the delay area in the RAM 3
21 constituting all delay circuits on the physical model sound source are cleared
22 and initialized to generate the music tone corresponding to the new key-on. If
23 the number of sound channels in the physical model sound source is plural, the
24 delay area in the RAM 3 constituting the delay circuit for the sound channel
25 attenuated most is cleared to mute the music tone of that sound channel. Then,

1 using the initialized delay area, the music tone corresponding to the new key-
2 on is generated.

3 Clearing the delay area in the RAM 3 is realized by writing data "0" to
4 that area, so that the music tone generation is unnaturally delayed by the time of
5 clearing. FIG. 41 shows a hardware constitution of a delay circuit that can
6 eliminate the wait time for clearing the delay area. As shown in FIG. 41, the
7 delay circuit is made up of two systems of delay means. The delay means of the
8 first delay system is composed of a multiplying means MU31, a delay means
9 DELAYa, and a multiplying means MU32 interconnected in series. The delay
10 means of the second delay system is composed of a multiplying means MU33, a
11 delay means DELAYb, and a multiplying means MU34 interconnected in
12 series. Input data INPUT is inputted in both the first and second delay systems.
13 The outputs of both of the delay systems are added by an adding means
14 AD31, and outputted as delay output data OUTPUT. The multiplying means
15 MU31 is provided with a multiplication coefficient INGAINa, the multiplying
16 means MU33 is provided with a multiplication coefficient INGAINb, the
17 multiplying means MU32 is provided with a multiplication coefficient
18 OUTGAINa, and the multiplying means MU34 is provided with a
19 multiplication coefficient OUTGAINb. As shown in FIG. 41, an input
20 controller is composed of the multiplying means MU31 and MU32. A mixer
21 (MIX) is composed of the multiplying means MU32 and MU34 and the adding
22 means AD31. In FIG. 41, the delay circuit is represented in hardware approach.
23 Actually, the delay circuit is implemented by software, namely a delay
24 processing program that uses the delay area in the RAM 3.

25 The following explains the operation of the delay circuit shown in FIG.
26 41 with reference to FIGS. 42A and 42B. FIG. 42A shows an equivalent circuit

1 means DELAYa. The delay input data is outputted via the selector 32 as output
2 data OUTPUT delayed by the predetermined time. If the multiplication
3 coefficient INPUTa and the multiplication coefficient OUTPUTa are set to "0"
4 and the multiplication coefficient INPUTb and the multiplication coefficient
5 OUTPUTb are set to "1", the input data INPUT is led by the selector 31 to the
6 delay means DELAYb and is delayed by a time corresponding to a delay
7 amount DLYb set by the delay means DELAYb. The delayed input data is
8 outputted via the selector 32 as output data OUTPUT delayed by the
9 predetermined time.

10 The first delay system and the second delay system can be switched to
11 each other in a toggle manner. Therefore, if the first delay system is in use for
12 example when a new key-on occurs, the multiplication coefficient between the
13 multiplication coefficient INPUTa and the multiplication coefficient
14 OUTPUTa in the first delay system is changed from "1" to "0". At the same
15 time, the multiplication coefficient between the multiplication coefficient
16 INPUTb and the multiplication coefficient OUTPUTb in the second delay
17 system is changed from "0" to "1". These changing operations allow the use of
18 the delay means DELAYb in the second delay system. Thus, it is ready to
19 generate the music tone corresponding to the new key-on. Because the
20 multiplication coefficient in the first delay system is changed to "0", data "0" is
21 written to the delay means DELAYa of the first delay system in one period of
22 music tone, thereby clearing this delay means.

23 The delay circuit shown in FIG. 42A is represented in hardware
24 approach. When the above-mentioned delay control is performed by software,
25 the selectors 31 and 32 need not be provided on the input side and the output
26 side. The operations equivalent to these selectors can be performed by

1 allocating a free delay area in the RAM 3 every time key-on occurs. When new
2 key-on occurs, the delay means of the delay system to which multiplication
3 coefficient "0" is set shifts by the delay length used so far by the write pointer
4 (or by the memory area allocated to the delay concerned) and is written with
5 data "0" to be cleared. The memory area may be kept in the wait state until the
6 same is allocated with key-on to be generated next. Preferably, a flag is set on
7 this memory area indicating that this area is free. Further, when new key-on
8 occurs, the delay system released by truncate processing may be cleared when
9 the load of the CPU is not heavy.

10 The delay circuit shown in FIG. 42B is obtained by replacing the selector
11 32 of the delay circuit shown in FIG. 42A by a mixer (MIX) 34. The delay
12 circuit of FIG. 42B can perform the same delay control as that of the delay
13 circuit shown in FIG. 42A. In the delay circuit shown in FIG. 42B, the delay
14 systems can be switched by the selector 33 and, at the same time, cross-fade
15 control can be performed in which the multiplication coefficients OUTGAINa
16 and OUTGAINb set, respectively, to the multipliers MU32 and MU34
17 constituting the mixer 34 are gradually switched from "1" to "0" or from "0" to
18 "1". Within one music tone period, gradual shift can be made from one music
19 tone to another.

20 In the delay circuit shown in FIG. 41, the first delay system and the
21 second delay system are always operated in parallel with the multiplication
22 coefficients INGAINa and INGAINb both set to "1" and, every time key-on
23 occurs, a delay amount DLY is set to the delay system other than the delay
24 system assigned to the preceding key-on to provide the pitch corresponding to
25 the new key-on. For example, if the first delay system is assigned to the last
26 key-on, a delay amount DLYb corresponding to the pressing key pitch is set to

1 the delay means DELAYb of the second delay system. At the same time, the
2 multiplication coefficient OUTGAINa of the first delay system is gradually
3 changed from "1" to "0" and the multiplication coefficient OUTGAINb is
4 gradually changed from "0" to "1". When the first delay system and the second
5 delay system are thus cross-fade controlled, the delay amount of the output data
6 OUTPUT outputted from the adding means AD31 substantially changes from
7 the delay amount DLYa to the delay amount DLYb smoothly. Namely,
8 portamento can be achieved. Further, a music tone of which pitch changes at
9 any pitch curve may be obtained by performing cross-fade control on the first
10 delay system and the second delay system alternately and repeatedly, and by
11 changing arbitrarily, every time cross-fade control is performed, the delay
12 amount DLY set to the delay means of the delay system of which multiplication
13 coefficient gradually changes to "1". Moreover, the first delay system and the
14 second delay system are used as delay circuits corresponding to different
15 sampling frequencies, and the delay amounts of these delay circuits are made
16 equal to each other. Besides, while a sum of the multiplication coefficient
17 INGAINa and the multiplication coefficient INGAINb becomes "1" and a sum
18 of the multiplication coefficient OUTGAINa and the multiplication coefficient
19 OUTGAINb becomes "1", each multiplication coefficient is controlled
20 appropriately. This mixes timbres based on different sampling frequencies,
21 thereby generating a music tone having a new timbre. If the signal amplitude of
22 a branch path in the physical model sound source becomes small, shift from the
23 preceding key-on to the current key-on may shift to the delay system having the
24 lower sampling frequency. When the shift has been completed, the delay
25 system of which assignment has been cleared can be assigned to the other delay
26 circuit.

1 The above-mentioned delay circuits are implemented by software by
2 using the delay areas set in the RAM 3. This is schematically illustrated in FIG.
3 43. As shown in the figure, a predetermined area in the RAM 3 is assigned to
4 the delay area. This delay area is divided into a plurality of delay areas to
5 provide unit delay areas (DELAY1a, DELAY1b, ..., DELAYA9, ..., DELAYn)
6 for constituting the delay means. These unit delay areas are allocated to the
7 delay means (DELAY1, ..., DELAYn). A flag area may be provided for each of
8 these unit delay areas. A free flag may be set to this flag area, indicating that
9 the unit delay area is not used as a delay means and hence free.

10 The following explains the allocation of the delay area for implementing
11 the delay circuit shown in FIG. 41 with reference to FIG. 43. It should be noted
12 that the physical model sound source has first delay circuit through the n-th
13 delay circuit. By the preceding key-on, the unit delay area DELAYa has been
14 allocated to the delay means of the first delay system of the first delay circuit
15 DELAY1 for example, and the delay amount of the unit delay area DELAY1a
16 is set to delay amount DLYi according to the pitch associated with the
17 preceding key-on. Further, by the preceding key-on, the unit delay area
18 DELAY9 has been allocated to the delay means of the first delay system of the
19 n-th delay circuit DELAYn for example, and the delay amount of the unit delay
20 area DELAY9 is set to delay amount DLYi according to the pitch associated
21 with the preceding key-on.

22 Next, when the current key-on occurs, the unit delay area DELAY1b is
23 allocated to the delay means of the second delay system of the first delay circuit
24 DELAY1 for example, and the delay amount of the unit delay area DELAY1a
25 is set to delay amount DLYk according to the pitch of the current key-on. By
26 the current key-on, the unit delay area DELAYn is allocated to the delay means

1 DELAY_n of the second delay system of the nth delay circuit for example, and
2 the delay amount of the unit delay area DELAY_n is set to delay amount DLY_k
3 according to the pitch associated with the current key-on. This can perform the
4 operation of the delay circuit shown in FIG. 41.

The constitution shown in FIG. 43 indicates that the physical model sound source has a single sound channel. FIG. 44 shows the allocation of the delay area for implementing the delay circuit when the physical model sound source has a plurality of sound channels. The following explains the operation of this constitution. When the unit delay area DELAY1a has been allocated to the delay means of the first delay system in the delay circuit DELAY1 of the first channel for example by the preceding key-on, the delay amount of the unit delay area DELAY1a is set to delay amount DLYp according to the pitch of the preceding key-on allocated to the first sound channel. Then, when the current key-on occurs and the unit delay area DELAY1b is allocated to the delay means of the second delay system in the delay circuit DELAY1 of the first sound channel for example, the delay amount of the unit delay area DELAY1a is set to delay amount DLYq according to the pitch associated with the current key-on allocated to the first sound channel. If the unit delay area DELAY9 has been allocated to the delay means of the first delay system in the delay circuit DELAYn of the second sound channel for example by the preceding key-on, the delay amount of the unit delay area DELAY9 is set to the delay amount DLYp according to the pitch associated with the preceding key-on. Then, if the unit delay area DELAYn is allocated to the delay means DELAYn of the second delay system of the second sound channel for example by the current key-on, the delay amount of the unit delay area DELAYn is set to the delay amount DLYq according to the pitch associated with the current key-on. This

1 arrangement allows execution of the operation of the delay circuit shown in
2 FIG. 41 if the physical model sound source has a plurality of sound channels.
3 In the constitutions of FIGS. 43 and 44, the unit delay area to be allocated to
4 each delay circuit may be previously determined in a fixed manner.
5 Alternatively, the allocation may be performed dynamically by checking, every
6 time key-on occurs, the free flag set to the unit delay area.

7 As described above, the inventive tone generating method uses a
8 hardware processor having a software module used to compute samples of a
9 waveform for generating a musical tone. The inventive method comprises the
10 steps of periodically providing a trigger signal at a relatively slow rate to define
11 a frame period between successive trigger signals, periodically providing a
12 sampling signal at a relatively fast rate such that a plurality of sampling signals
13 occur within one frame period, operating the hardware processor resettable in
14 response to a trigger signal and operable in response to each sampling signal to
15 periodically execute the software module for successively computing a number
16 of samples of the waveform within one frame, and converting each of the
17 samples into a corresponding analog signal in response to each sampling signal
18 to thereby generate the musical tones. The step of operating includes delaying
19 step using a pair of memory regions for imparting a delay to the waveform to
20 determine a pitch of the musical tone according to the performance information.
21 The delay step successively writes the samples of the waveform of one musical
22 tone into addresses of one of the memory regions, and successively reads the
23 samples from addresses of the same memory region to thereby create the delay.
24 The delay step responds when the hardware processor is reset so that said one
25 musical tone is switched to another musical tone for successively writing the
26 samples of the waveform of said another musical tone into addresses of the

1 other memory region and successively reading the samples from addresses of the
2 same memory region to thereby create the delay while clearing the one memory
3 region to prepare for a further musical tone.

4 Described so far is the software sound source that practices the second
5 preferred embodiment of the invention on a personal computer. In the
6 computer system, this sound source software can be handled as either
7 application software or device drive software, for example. The way by which
8 the sound source software is to be handled may be appropriately determined
9 according to the system configuration or the operation system OS used.

10 The sound source software or the capabilities thereof may be
11 incorporated in another software program such as amusement software,
12 karaoke software, or automatic play and accompaniment software. Also this
13 software may be directly incorporated in the operation system OS. The
14 software according to the present invention can be supplied in a machine-
15 readable disk media such as a floppy disk, a magneto-optical disk, and a CD-
16 ROM or a memory card. Further, the software may be added by means of a
17 semiconductor memory chip (typically ROM) which is inserted in a computer
18 unit. Alternatively, the sound source software associated with the present
19 invention may be distributed through the network I/F 11.

20 The above description has been made by using the application on a
21 personal computer for example. Application to amusement equipment such as
22 game and karaoke, electronic equipment, and general-purpose electrical
23 equipment is also practical. In addition, application to a sound source board
24 and a sound source unit is practical. Moreover, application to a sound source
25 machine based on software processing using dedicated MPU (DS) is practical.
26 In this case, if the processing capacity of the MPU is high, the sampling

1 frequency can be raised, thereby multiplying the sampling frequency by n when
2 high-precision waveform output is required. Further, when a plurality of sound
3 channels are used on the sound source, variable control on the sampling
4 frequency and skip control on the computation portion that can be skipped in
5 the computation algorithm may be performed according to the number of
6 channels being sounded. In this case, different sampling frequencies may be
7 set to different performance parts or MIDI channels. Still further, in the
8 above-mentioned embodiment, the sampling frequency of the CODEC is fixed.
9 It will be apparent that this sampling frequency is variable. The sampling
10 frequency is made variable by inserting the processing circuit for matching the
11 sampling frequencies between the waveform output buffer WAVEBUF and the
12 CODEC (DAC) by typically oversampling, downsampling, or data
13 interpolation.

14 The present invention is applicable to a software sound source in which
15 the CPU operates in synchronization with the sampling frequency to
16 periodically execute the software module for successively computing
17 waveform samples. For example, the CPU conducts an interrupt for computing
18 one sample at a period of $1/(n \times f_s)$ where n denotes a number of tones and f_s
19 denotes a sampling frequency. Further, the invention is applicable to a
20 hardware sound source using an LSI chip in order to reduce load of ALU and in
21 order to use resources of LSI chip for other tasks than tone generation.

22 As described and according to the present invention, music tone
23 waveform generating blocks indicated by a preset algorithm are assigned to
24 selected sound channels, the assigned music tone waveform generating blocks
25 are combined by the algorithm, and music tone waveform generating
26 computation is performed to generate music tone waveform data.

1 Consequently, the number of music waveform generating blocks for the sound
2 channels may be arbitrarily changed before sounding assignment is made. This
3 novel constitution allows, according to the capacity of a music waveform data
4 generating means, flexible adjustment of the load state of the music waveform
5 data generating means and the quality of the music waveform data to be
6 generated.

7 The music tone waveform generating blocks indicated by an algorithm
8 set according to the timbre of the music tone are assigned to the selected sound
9 channels. The assigned music tone waveform generating blocks are combined
10 by the algorithm to perform music tone waveform generating computation so as
11 to generate the music tone waveform data.

12 Preferably, in setting timbres by a timbre setting means, if the number of
13 music tone waveform generating blocks is set to a performance part concerned
14 by a means for setting number of blocks, the timbre set to that performance part
15 is changed to a timbre defined by music tone waveform generating blocks
16 within that number of blocks. This novel constitution further enhances the
17 above-mentioned effect.

18 Preferably, during the music tone waveform generating computation in
19 the sound channel, the number of music tone waveform generating blocks
20 assigned to that sound channel is changed according to a predetermined
21 condition. Consequently, during sounding, the load state of the music tone
22 waveform data generating means and the quality of the music waveform data to
23 be generated may be changed flexibly according to the capacity of that music
24 tone waveform generating means.

25 Further, according to the present invention, in a computer equipment
26 which often executes a plurality of tasks such as word processing and network

1 communication in addition to music performance, occurrence of troubles such
2 as an interrupted music tone can be reduced when the CPU power is allocated
3 to the tasks not associated with music performance during processing of the
4 software sound source. In other words, more tasks can be undertaken during
5 the execution of sound source processing.

6 Since the present invention is constituted as described above, when the
7 CPU load is high, the sampling frequency can be lowered, thereby generating
8 tone waveform data that prevents the interruption of a music tone. When the
9 CPU load is low, a higher sampling frequency than the normal sampling
10 frequency can be used, thereby generating high-precision tone waveform data.
11 In this case, the number of sound channels may be changed instead of changing
12 the sampling frequency.

13 If a particular condition is satisfied, corresponding computational
14 operations are skipped, so that efficient computation can be performed, thereby
15 preventing the CPU load from getting extremely high. Consequently, the tone
16 waveform data can be generated that prevents the sounding of a music tone
17 from being interrupted. Further, the efficient computation allows the use of the
18 higher sampling frequency than the conventional sampling frequency, resulting
19 in high-precision tone waveform data.

20 While the preferred embodiments of the present invention have been
21 described using specific terms, such description is for illustrative purposes only,
22 and it is to be understood that changes and variations may be made without
23 departing from the spirit or scope of the appended claims.